Package ‘psychmeta’

August 26, 2022

Type Package
Title Psychometric Meta-Analysis Toolkit
Version 2.6.5
Date 2022-08-25

BugReports https://github.com/psychmeta/psychmeta/issues

Description
Tools for computing bare-bones and psychometric meta-analyses and for generating psychometric data for use in meta-analysis simulations. Supports bare-bones, individual-correction, and artifact-distribution methods for meta-analyzing correlations and d values. Includes tools for converting effect sizes, computing sporadic artifact corrections, reshaping meta-analytic databases, computing multivariate corrections for range variation, and more. Bugs can be reported to <https://github.com/psychmeta/psychmeta/issues> or <issues@psychmeta.com>.

License GPL (>= 3)
Depends R (>= 3.5.0)
Encoding UTF-8
LazyData true
VignetteBuilder knitr
Imports stats, boot, metafor, ggplot2, progress, curl, dplyr, tibble, tidyr (>= 1.0.0), rlang, purrr
Suggests MASS, mvtnorm, nor1mix, bib2df, rmarkdown, knitr, stringi, cli, crayon, testthat (>= 2.1.0)

RoxygenNote 7.2.1.9000

NeedsCompilation no

Author Jeffrey A. Dahlke [aut, cre], Brenton M. Wiernik [aut], Wesley Gardiner [ctb] (Unit tests), Michael T. Brannick [ctb] (Testing), Jack Kostal [ctb] (Code for reshape_mat2dat function), Sean Potter [ctb] (Testing: Code for cumulative and leave1out plots), John Sakaluk [ctb] (Code for funnel and forest plots), Yuejia (Mandy) Teng [ctb] (Testing)
R topics documented:

psychmeta-package .................................................. 4
.ma_r_order2 .......................................................... 7
.tau_squared_m_solver .............................................. 7
.tau_squared_r_solver ............................................... 8
adjust_n_d .............................................................. 9
adjust_n_r .............................................................. 10
anova.ma_psychmeta ............................................... 11
composite_d_matrix .................................................. 12
composite_d_scalar .................................................. 13
composite_rel_matrix ................................................ 14
composite_rel_scalar ............................................... 15
composite_r_matrix .................................................. 16
composite_r_scalar .................................................. 17
composite_u_matrix .................................................. 19
composite_u_scalar .................................................. 20
compute_alpha ......................................................... 21
compute_dmod ........................................................ 21
compute_dmod_npar ................................................... 24
compute_dmod_par .................................................... 27
conf.limits.nc.chisq .................................................. 29
confidence ............................................................ 31
confidence_r ........................................................... 32
confint ................................................................. 32
control_intercor ....................................................... 33
control_psychmeta .................................................... 35
convert_es .............................................................. 38
convert_ma ............................................................. 39
correct_d ............................................................ 41
correct_d_bias ........................................................ 43
correct_glass_bias .................................................... 45
correct_matrix_mvrr .................................................. 46
correct_means_mvrr ................................................... 47
correct_r .............................................................. 48
correct_r_bias ........................................................ 52
correct_r_coarseness ................................................. 53
correct_r_dich ........................................................ 54
correct_r_split ........................................................ 55
create_ad .............................................................. 56
create_ad_group ........................................................ 62
create_ad_tibble ....................................................... 64
credibility ............................................................ 68
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>data_d_bb_multi</td>
<td>69</td>
</tr>
<tr>
<td>data_d_meas_multi</td>
<td>69</td>
</tr>
<tr>
<td>data_r_bvdr</td>
<td>70</td>
</tr>
<tr>
<td>data_r_bvrr</td>
<td>70</td>
</tr>
<tr>
<td>data_r_gonzalezmulne_2014</td>
<td>71</td>
</tr>
<tr>
<td>data_r_mcdaniel_1994</td>
<td>71</td>
</tr>
<tr>
<td>data_r_mcleod_2007</td>
<td>72</td>
</tr>
<tr>
<td>data_r_meas</td>
<td>72</td>
</tr>
<tr>
<td>data_r_meas_multi</td>
<td>73</td>
</tr>
<tr>
<td>data_r_oh_2009</td>
<td>73</td>
</tr>
<tr>
<td>data_r_roth_2015</td>
<td>74</td>
</tr>
<tr>
<td>data_r_uvdr</td>
<td>75</td>
</tr>
<tr>
<td>data_r_uvrr</td>
<td>75</td>
</tr>
<tr>
<td>estimate_artifacts</td>
<td>76</td>
</tr>
<tr>
<td>estimate_length_sb</td>
<td>80</td>
</tr>
<tr>
<td>estimate_prod</td>
<td>81</td>
</tr>
<tr>
<td>estimate_q_dist</td>
<td>83</td>
</tr>
<tr>
<td>estimate_rel_dist</td>
<td>83</td>
</tr>
<tr>
<td>estimate_rel_sb</td>
<td>84</td>
</tr>
<tr>
<td>estimate_u</td>
<td>85</td>
</tr>
<tr>
<td>estimate_var_artifacts</td>
<td>87</td>
</tr>
<tr>
<td>estimate_var_rho_int</td>
<td>97</td>
</tr>
<tr>
<td>estimate_var_rho_tsa</td>
<td>99</td>
</tr>
<tr>
<td>estimate_var_tsa</td>
<td>108</td>
</tr>
<tr>
<td>filter_ma</td>
<td>112</td>
</tr>
<tr>
<td>format_num</td>
<td>113</td>
</tr>
<tr>
<td>generate_bib</td>
<td>115</td>
</tr>
<tr>
<td>generate_directory</td>
<td>117</td>
</tr>
<tr>
<td>get_stuff</td>
<td>117</td>
</tr>
<tr>
<td>heterogeneity</td>
<td>123</td>
</tr>
<tr>
<td>limits_tau</td>
<td>125</td>
</tr>
<tr>
<td>limits_tau2</td>
<td>126</td>
</tr>
<tr>
<td>lm_mat</td>
<td>127</td>
</tr>
<tr>
<td>ma_d</td>
<td>130</td>
</tr>
<tr>
<td>ma_d_order2</td>
<td>143</td>
</tr>
<tr>
<td>ma_generic</td>
<td>144</td>
</tr>
<tr>
<td>ma_r</td>
<td>146</td>
</tr>
<tr>
<td>ma_r_order2</td>
<td>164</td>
</tr>
<tr>
<td>merge_simdat_d</td>
<td>166</td>
</tr>
<tr>
<td>merge_simdat_r</td>
<td>166</td>
</tr>
<tr>
<td>metabulate</td>
<td>167</td>
</tr>
<tr>
<td>metabulate_rmd_helper</td>
<td>171</td>
</tr>
<tr>
<td>metareg</td>
<td>173</td>
</tr>
<tr>
<td>mix_dist</td>
<td>174</td>
</tr>
<tr>
<td>mix_matrix</td>
<td>176</td>
</tr>
<tr>
<td>mix_r_2group</td>
<td>177</td>
</tr>
<tr>
<td>plot_forest</td>
<td>178</td>
</tr>
<tr>
<td>plot_funnel</td>
<td>179</td>
</tr>
</tbody>
</table>
psychmeta-package

psychmeta: Psychometric meta-analysis toolkit

Description

Overview of the psychmeta package.
**psychmeta-package**

**Details**

The `psychmeta` package provides tools for computing bare-bones and psychometric meta-analyses and for generating psychometric data for use in meta-analysis simulations. Currently, `psychmeta` supports bare-bones, individual-correction, and artifact-distribution methods for meta-analyzing correlations and \( d \) values. Please refer to the overview tutorial vignette for an introduction to `psychmeta`’s functions and workflows.

**Running a meta-analysis**

The main functions for conducting meta-analyses in `psychmeta` are `ma_r` for correlations and `ma_d` for \( d \) values. These functions take meta-analytic dataframes including effect sizes and sample sizes (and, optionally, study labels, moderators, construct and measure labels, and psychometric artifact information) and return the full results of psychometric meta-analyses for all of the specified variable pairs. Examples of correctly formatted meta-analytic datasets for `ma` functions are `data_r_roth_2015`, `data_r_gonzalezmule_2014`, and `data_r_mcdaniel_1994`. Individual parts of the meta-analysis process can also be run separately; these functions are described in detail below.

**Preparing a database for meta-analysis**

The `convert_es` function can be used to convert a variety of effect sizes to either correlations or \( d \) values. Sporadic psychometric artifacts, such as artificial dichotomization or uneven splits for a truly dichotomous variable, can be individually corrected using `correct_r` and `correct_d`. These functions can also be used to compute confidence intervals for observed, converted, and corrected effect sizes. ‘Wide’ meta-analytic coding sheets can be reformatted to the ‘long’ data frames used by `psychmeta` with `reshape_wide2long`. A correlation matrix and accompanying vectors of information can be similarly reformatted using `reshape_mat2dat`.

**Meta-analytic models**

`psychmeta` can compute barebones meta-analyses (no corrections for psychometric artifacts), as well as models correcting for measurement error in one or both variables, univariate direct (Case II) range restriction, univariate indirect (Case IV) range restriction, bivariate direct range restriction, bivariate indirect (Case V) range restriction, and multivariate range restriction. Artifacts can be corrected individually or using artifact distributions. Artifact distribution corrections can be applied using either Schmidt and Hunter’s (2015) interactive method or Taylor series approximation models. Meta-analyses can be computed using various weights, including sample size (default for correlations), inverse variance (computed using either sample or mean effect size; error based on mean effect size is the default for \( d \) values), and weight methods imported from `metafor`.

**Preparing artifact distributions meta-analyses**

For individual-corrections meta-analyses, reliability and range restriction (u) values should be supplied in the same data frame as the effect sizes and sample sizes. Missing artifact data can be imputed using either bootstrap or other imputation methods. For artifact distribution meta-analyses, artifact distributions can be created automatically by `ma_r` or `ma_d` or manually by the `create_ad` family of functions.
Moderator analyses

Subgroup moderator analyses are run by supplying a moderator matrix to the `ma_r` or `ma_d` families of functions. Both simple and fully hierarchical moderation can be computed. Subgroup moderator analysis results are shown by passing an `ma_obj` to `print()`. Meta-regression analyses can be run using `metareg`.

Reporting results and supplemental analyses

Meta-analysis results can be viewed by passing an `ma` object to `summary`. Bootstrap confidence intervals, leave one out analyses, and other sensitivity analyses are available in `sensitivity`. Supplemental heterogeneity statistics (e.g., $Q$, $I^2$) can be computed using `heterogeneity`. Meta-analytic results can be converted between the $r$ and $d$ metrics using `convert_ma`. Each `ma_obj` contains a `metafor escalc` object in `ma$...$escalc` that can be passed to `metafor`’s functions for plotting, publication/availability bias, and other supplemental analyses. Second-order meta-analyses of correlations can be computed using `ma_r_order2`. Example second-order meta-analysis datasets from Schmidt and Oh (2013) are available. Tables of meta-analytic results can be written as markdown, Word, HTML, or PDF files using the `metabulate` function, which exports near publication-quality tables that will typically require only minor customization by the user.

Simulating psychometric meta-analyses

`psychmeta` can be used to run Monte Carlo simulations for different meta-analytic models. `simulate_r_sample` and `simulate_d_sample` simulate samples of correlations and $d$ values, respectively, with measurement error and/or range restriction artifacts. `simulate_r_database` and `simulate_d_database` can be used to simulate full meta-analytic databases of sample correlations and $d$ values, respectively, with artifacts. Example datasets fitting different meta-analytic models simulated using these functions are available (`data_r_meas`, `data_r_uvdrr`, `data_r_uvirr`, `data_r_bvdrr`, `data_r_bvirr`, `data_r_meas_multi`, and `data_d_meas_multi`). Additional simulation functions are also available.

Author(s)

Maintainer: Jeffrey A. Dahlke <jdahlke@humrro.org>

Authors:

• Brenton M. Wiernik <brenton@psychmeta.com>

Other contributors:

• Wesley Gardiner (Unit tests) [contributor]
• Michael T. Brannick (Testing) [contributor]
• Jack Kostal (Code for reshape_mat2dat function) [contributor]
• Sean Potter (Testing; Code for cumulative and leave1out plots) [contributor]
• John Sakaluk (Code for funnel and forest plots) [contributor]
• Yuejia (Mandy) Teng (Testing) [contributor]
See Also

Useful links:

- Report bugs at https://github.com/psychmeta/psychmeta/issues

---

.ma_r_order2

Internal function for computing individual-correction meta-analyses of correlations

---

Description

Internal function for computing individual-correction meta-analyses of correlations

Usage

.ma_r_order2(data, type = "all", run_lean = FALSE, ma_arg_list)

Arguments

data Data frame of individual-correction information.
type Type of correlation to be meta-analyzed: "ts" for true score, "vgx" for validity generalization with "X" as the predictor, "vgy" for validity generalization with "X" as the predictor, and "all" for the complete set of results.
run_lean If TRUE, the meta-analysis will not generate a data object. Meant to speed up bootstrap analyses that do not require supplemental output.
ma_arg_list List of arguments to be used in the meta-analysis function.

Value

A meta-analytic table and a data frame.

---

.tau_squared_m_solver tau_m_squared Solver

---

Description

Function to solve for tau_m_squared (outlier-robust estimator of tau_squared based on absolute deviations from median)

Usage

.tau_squared_m_solver(Q_m, w_i, k)
Arguments

Q_r
The Q_r statistic.

wi
Vector of inverse within-study sampling variances.

k
The number of effect sizes included in the meta-analysis.

Value

tau_r_squared

Author(s)

Lifeng Lin and Haitao Chu

Description

Function to solve for tau_r_squared (outlier-robust estimator of tau_squared based on absolute deviations from mean)

Usage

.tau_squared_r_solver(Q_r, wi)

Arguments

Q_r
The Q_r statistic.

wi
Vector of inverse within-study sampling variances.

Value

tau_r_squared

Author(s)

Lifeng Lin and Haitao Chu
**adjust_n_d**  
*Adjusted sample size for a non-Cohen d value for use in a meta-analysis of Cohen’s d values*

**Description**

This function is used to convert a non-Cohen $d$ value (e.g., Glass’ $\Delta$) to a Cohen’s $d$ value by identifying the sample size of a Cohen’s $d$ that has the same standard error as the non-Cohen $d$. This function permits users to account for the influence of sporadic corrections on the sampling variance of $d$ prior to use in a meta-analysis.

**Usage**

```r
adjust_n_d(d, var_e, p = NA)
```

**Arguments**

- **d**: Vector of non-Cohen $d$ standardized mean differences.
- **var_e**: Vector of error variances of standardized mean differences.
- **p**: Proportion of participants in a study belonging to one of the two groups being contrasted.

**Details**

The adjusted sample size is computed as:

$$n_{\text{adjusted}} = \frac{d^2 p(1 - p) + 2}{2p(1 - p)var_e}$$

**Value**

A vector of adjusted sample sizes.

**References**


**Examples**

```r
adjust_n_d(d = 1, var_e = .03, p = NA)
```
**Description**

This function is used to compute the adjusted sample size of a non-Pearson correlation (e.g., a tetrachoric correlation) based on the correlation and its estimated error variance. This function allows users to adjust the sample size of a correlation corrected for sporadic artifacts (e.g., unequal splits of dichotomous variables, artificial dichotomization of continuous variables) prior to use in a meta-analysis.

**Usage**

`adjust_n_r(r, var_e)`

**Arguments**

- `r`: Vector of correlations.
- `var_e`: Vector of error variances.

**Details**

The adjusted sample size is computed as:

\[
    n_{adjusted} = \frac{(r^2 - 1)^2 + \text{var}_e}{\text{var}_e}
\]

**Value**

A vector of adjusted sample sizes.

**References**


**Examples**

```r
adjust_n_r(r = .3, var_e = .01)
```
Description

This function computes Wald-type pairwise comparisons for each level of categorical moderators for an `ma_psychmeta` object, as well as an omnibus one-way ANOVA test (equal variance not assumed).

Currently, samples across moderator levels are assumed to be independent.

Usage

```r
## S3 method for class 'ma_psychmeta'
anova(
  object,
  ..., 
  analyses = "all",
  moderators = NULL,
  L = NULL,
  ma_obj2 = NULL,
  ma_method = c("bb", "ic", "ad"),
  correction_type = c("ts", "vgx", "vgy"),
  conf_level = NULL
)
```

Arguments

- **object**: A psychmeta meta-analysis object.
- **...**: Additional arguments.
- **analyses**: Which analyses to test moderators for? Can be either "all" to test moderators for all meta-analyses in the object (default) or a list containing one or more of the arguments `construct`, `construct_pair`, `pair_id`, `k_min`, and `N_min`. See `filter_ma()` for details. Note that `analysis_id` should not be used. If `k_min` is not supplied, it is set to 2.
- **moderators**: A character vector of moderators to test. If NULL, all categorical moderators are tested.
- **L**: A named list with with elements specifying set of linear contrasts for each variable in moderators. (Not yet implemented.)
- **ma_obj2**: A second psychmeta meta-analysis object to compare to `object` (Not yet implemented.)
- **ma_method**: Meta-analytic methods to be included. Valid options are: "bb", "ic", and "ad".
- **correction_type**: Types of meta-analytic corrections to be included. Valid options are: "ts", "vgx", and "vgy".
- **conf_level**: Confidence level to define the width of confidence intervals (defaults to value set when object was fit).
Value

An object of class anova.ma_psychmeta. A tibble with a row for each construct pair in object and a column for each moderator tested. Cells lists of contrasts tested.

Note

Currently, only simple (single) categorical moderators (one-way ANOVA) are supported.

Examples

```r
ma_obj <- ma_r(rxyi, n, construct_x = x_name, construct_y = y_name, moderators = moderator, data = data_r_meas_multi)
anova(ma_obj)
```

composite_d_matrix

Matrix formula to estimate the standardized mean difference associated with a weighted or unweighted composite variable

Description

This function is a wrapper for composite_r_matrix that converts \(d\) values to correlations, computes the composite correlation implied by the \(d\) values, and transforms the result back to the \(d\) metric.

Usage

```r
composite_d_matrix(d_vec, r_mat, wt_vec, p = 0.5)
```

Arguments

- `d_vec` Vector of standardized mean differences associated with variables in the composite to be formed.
- `r_mat` Correlation matrix from which the composite is to be computed.
- `wt_vec` Weights to be used in forming the composite (by default, all variables receive equal weight).
- `p` The proportion of cases in one of the two groups used to compute the standardized mean differences.

Details

The composite \(d\) value is computed by converting the vector of \(d\) values to correlations, computing the composite correlation (see composite_r_matrix), and transforming that composite back into the \(d\) metric. See "Details" of composite_r_matrix for the composite computations.

Value

The estimated standardized mean difference associated with the composite variable.
Examples

```
composite_d_matrix(d_vec = c(1, 1), r_mat = matrix(c(1, .7, .7, 1), 2, 2),
                  wt_vec = c(1, 1), p = .5)
```

\[ \text{composite}_d_{\text{scalar}} \]  
Scalar formula to estimate the standardized mean difference associated with a composite variable

Description

This function estimates the \( d \) value of a composite of \( X \) variables, given the mean \( d \) value of the individual \( X \) values and the mean correlation among those variables.

Usage

```
composite_d_scalar(
  mean_d,
  mean_intercor,
  k_vars,
  p = 0.5,
  partial_intercor = FALSE
)
```

Arguments

- `mean_d`: The mean standardized mean differences associated with variables in the composite to be formed.
- `mean_intercor`: The mean correlation among the variables in the composite.
- `k_vars`: The number of variables in the composite.
- `p`: The proportion of cases in one of the two groups used to compute the standardized mean differences.
- `partial_intercor`: Logical scalar determining whether the intercor represents the partial (i.e., within-group) correlation among variables (TRUE) or the overall correlation between variables (FALSE).

Details

There are two different methods available for computing such a composite, one that uses the partial intercorrelation among the \( X \) variables (i.e., the average within-group correlation) and one that uses the overall correlation among the \( X \) variables (i.e., the total or mixture correlation across groups).

If a partial correlation is provided for the interrelationships among variables, the following formula is used to estimate the composite \( d \) value:

\[
 d_X = \frac{\bar{d}_x k}{\sqrt{\bar{\rho}_{xx} k^2 + (1 - \bar{\rho}_{xx}) k}}
\]
where $d_X$ is the composite $d$ value, $\bar{d}_{x_i}$ is the mean $d$ value, $\bar{\rho}_{x_i}$ is the mean intercorrelation among the variables in the composite, and $k$ is the number of variables in the composite. Otherwise, the composite $d$ value is computed by converting the mean $d$ value to a correlation, computing the composite correlation (see composite_r_scalar for formula), and transforming that composite back into the $d$ metric.

**Value**

The estimated standardized mean difference associated with the composite variable.

**References**


**Examples**

```r
composite_d_scalar(mean_d = 1, mean_intercor = .7, k_vars = 2, p = .5)
```

**composite_rel_matrix**  
Matrix formula to estimate the reliability of a weighted or unweighted composite variable

**Description**

This function computes the reliability of a variable that is a weighted or unweighted composite of other variables.

**Usage**

```r
composite_rel_matrix(rel_vec, r_mat, sd_vec, wt_vec = rep(1, length(rel_vec)))
```

**Arguments**

- `rel_vec`: Vector of reliabilities associated with variables in the composite to be formed.
- `r_mat`: Correlation matrix from which the composite is to be computed.
- `sd_vec`: Vector of standard deviations associated with variables in the composite to be formed.
- `wt_vec`: Weights to be used in forming the composite (by default, all variables receive equal weight).
composite_rel_scalar

Details

This function treats measure-specific variance as reliable.

The Mosier composite formula is computed as:

$$\rho_{XX} = \frac{w^T (r \circ s) + w^T Sw - w^T s}{w^T Sw}$$

where $\rho_{XX}$ is a composite reliability estimate, $r$ is a vector of reliability estimates, $w$ is a vector of weights, $S$ is a covariance matrix, and $s$ is a vector of variances (i.e., the diagonal elements of $S$).

Value

The estimated reliability of the composite variable.

References


Examples

```r
composite_rel_matrix(rel_vec = c(.8, .8),
r_mat = matrix(c(1, .4, .4, 1), 2, 2),
sd_vec = c(1, 1))
```

```r
composite_rel_scalar(mean_rel, mean_intercor, k_vars)
```

Description

This function computes the reliability of a variable that is a unit-weighted composite of other variables.

Usage

```r
composite_rel_scalar(mean_rel, mean_intercor, k_vars)
```

Arguments

- **mean_rel**: The mean reliability of variables in the composite.
- **mean_intercor**: The mean correlation among the variables in the composite.
- **k_vars**: The number of variables in the composite.
Details

The Mosier composite formula is computed as:

\[ \rho_{XX} = \bar{\rho}_{x_ix_i}k + k(k - 1) \bar{\rho}_{x_ix_j} \]

where \( \bar{\rho}_{x_ix_i} \) is the mean reliability of variables in the composite, \( \bar{\rho}_{x_ix_j} \) is the mean intercorrelation among variables in the composite, and \( k \) is the number of variables in the composite.

Value

The estimated reliability of the composite variable.

References


Examples

```r
composite_rel_scalar(mean_rel = .8, mean_intercor = .4, k_vars = 2)
```

---

**composite_r_matrix**

*Matrix formula to estimate the correlation between two weighted or unweighted composite variables*

Description

This function computes the weighted (or unweighted, by default) composite correlation between a set of X variables and a set of Y variables.

Usage

```r
composite_r_matrix(
  r_mat,
  x_col,
  y_col,
  wt_vec_x = rep(1, length(x_col)),
  wt_vec_y = rep(1, length(y_col))
)
```
**composite_r_scalar**

**Arguments**

- **r_mat**: Correlation matrix from which composite correlations are to be computed.
- **x_col**: Column indices of variables from 'r_mat' in the X composite (specify a single variable if Y is an observed variable rather than a composite).
- **y_col**: Column indices of variables from 'r_mat' in the Y composite (specify a single variable if Y is an observed variable rather than a composite).
- **wt_vec_x**: Weights to be used in forming the X composite (by default, all variables receive equal weight).
- **wt_vec_y**: Weights to be used in forming the Y composite (by default, all variables receive equal weight).

**Details**

This is computed as:

$$
\rho_{XY} = \frac{w_X^T R_{XY} w_Y}{\sqrt{(w_X^T R_{XX} w_X) (w_Y^T R_{YY} w_Y)}}
$$

where $\rho_{XY}$ is the composite correlation, $w$ is a vector of weights, and $R$ is a correlation matrix. The subscripts of $w$ and $R$ indicate the variables indexed within the vector or matrix.

**Value**

A composite correlation

**References**


**Examples**

```r
composite_r_scalar(mean_rxy = .3, k_vars_x = 4, mean_intercor_x = .4)  
R <- reshape_vec2mat(.4, order = 5)  
R[-1,1] <- R[1,-1] <- .3  
composite_r_matrix(r_mat = R, x_col = 2:5, y_col = 1)
```

---

**Description**

This function estimates the correlation between a set of X variables and a set of Y variables using a scalar formula.
Usage

\texttt{composite_r_scalar(}
  \texttt{mean_rxy,}
  \texttt{k_vars_x = \texttt{NULL},}
  \texttt{mean_intercor_x = \texttt{NULL},}
  \texttt{k_vars_y = \texttt{NULL},}
  \texttt{mean_intercor_y = \texttt{NULL}}
\texttt{)}

Arguments

- \texttt{mean_rxy}  Mean correlation between sets of X and Y variables.
- \texttt{k_vars_x}  Number of X variables.
- \texttt{mean_intercor_x}  Mean correlation among X variables.
- \texttt{k_vars_y}  Number of Y variables.
- \texttt{mean_intercor_y}  Mean correlation among Y variables.

Details

The formula to estimate a correlation between one composite variable and one external variable is:

\[ \rho_{XY} = \frac{\bar{\rho}_{x,y}}{\sqrt{\frac{1}{k_x} + \frac{k_x - 1}{k_x} \bar{\rho}_{x,x}}} \]

and the formula to estimate the correlation between two composite variables is:

\[ \rho_{XY} = \frac{\bar{\rho}_{x,y}}{\sqrt{\frac{1}{k_x} + \frac{k_x - 1}{k_x} \bar{\rho}_{x,x}} \sqrt{\frac{1}{k_y} + \frac{k_y - 1}{k_y} \bar{\rho}_{y,y}}} \]

where \( \bar{\rho}_{x,y} \) and \( \bar{\rho}_{x,y} \) are mean correlations between the x variables and the y variable(s), \( \bar{\rho}_{x,x} \) is the mean correlation among x variables, \( \bar{\rho}_{y,y} \) is the mean correlation among y variables, \( k_x \) is the number of x variables, and \( k_y \) is the number of y variables.

Value

A vector of composite correlations

References


Examples

```r
## Composite correlation between 4 variables and an outside variable with which
## the four variables correlate .3 on average:
composite_r_scalar(mean_rxy = .3, k_vars_x = 4, mean_intercor_x = .4)

## Correlation between two composites:
composite_r_scalar(mean_rxy = .3, k_vars_x = 2, mean_intercor_x = .5,
                  k_vars_y = 2, mean_intercor_y = .2)
```

**composite_u_matrix**  
*Matrix formula to estimate the u ratio of a composite variable*

**Description**

This function estimates the u ratio of a composite variable when at least one matrix of correlations (restricted or unrestricted) among the variables is available.

**Usage**

```r
composite_u_matrix(
  ri_mat = NULL,
  ra_mat = NULL,
  u_vec,
  wt_vec = rep(1, length(u_vec)),
  sign_r_vec = 1
)
```

**Arguments**

- **ri_mat**: Range-restricted correlation matrix from which the composite is to be computed (if NULL, ri_mat is estimated from ra_mat).
- **ra_mat**: Unrestricted correlation matrix from which the composite is to be computed (if NULL, ra_mat is estimated from ri_mat).
- **u_vec**: Vector of u ratios associated with variables in the composite to be formed.
- **wt_vec**: Weights to be used in forming the composite (by default, all variables receive equal weight).
- **sign_r_vec**: The signs of the relationships between the variables in the composite and the variable by which range restriction was induced.

**Details**

This is computed as:

\[
\hat{u}_{\text{composite}} = \sqrt{\frac{(w \odot u)^T R_i (w \odot u)}{w^T R_a w}}
\]

where \( \hat{u}_{\text{composite}} \) is the composite u ratio, \( u \) is a vector of u ratios, \( R_i \) is a range-restricted correlation matrix, \( R_a \) is an unrestricted correlation matrix, and \( w \) is a vector of weights.
Value
The estimated $u$ ratio of the composite variable.

Examples

```r
composite_u_matrix(ri_mat = matrix(c(1,.3,.3,1), 2, 2), u_vec = c(.8,.8))
```

---

**composite_u_scalar**  
*Scalar formula to estimate the $u$ ratio of a composite variable*

Description

This function provides an approximation of the $u$ ratio of a composite variable based on the $u$ ratios of the component variables, the mean restricted intercorrelation among those variables, and the mean unrestricted correlation among those variables. If only one of the mean intercorrelations is known, the other will be estimated using the bivariate indirect range-restriction formula. This tends to compute a conservative estimate of the $u$ ratio associated with a composite.

Usage

```r
composite_u_scalar(mean_ri = NULL, mean_ra = NULL, mean_u, k_vars)
```

Arguments

- `mean_ri`: The mean range-restricted correlation among variables in the composite.
- `mean_ra`: The mean unrestricted correlation among variables in the composite.
- `mean_u`: The mean $u$ ratio of variables in the composite.
- `k_vars`: The number of variables in the composite.

Details

This is computed as:

$$u_{\text{composite}} = \sqrt{\frac{\bar{\rho}_i \bar{u}^2 k(k-1) + k\bar{u}^2}{\bar{\rho}_a k(k-1) + k}}$$

where $u_{\text{composite}}$ is the composite $u$ ratio, $\bar{u}$ is the mean univariate $u$ ratio, $\bar{\rho}_i$ is the mean restricted correlation among variables, $\bar{\rho}_a$ is the mean unrestricted correlation among variables, and $k$ is the number of variables in the composite.

Value

The estimated $u$ ratio of the composite variable.

Examples

```r
composite_u_scalar(mean_ri = .3, mean_ra = .4, mean_u = .8, k_vars = 2)
```
compute_alpha

Description

Compute coefficient alpha

Usage

compute_alpha(sigma = NULL, data = NULL, standardized = FALSE, ...)

Arguments

sigma  Covariance matrix (must be supplied if data argument is not supplied)
data   Data matrix or data frame (must be supplied if sigma argument is not supplied)
standardized Logical scalar determining whether alpha should be computed from an unstandardized covariance matrix (TRUE) or a correlation matrix (FALSE).
...    Additional arguments to be passed to cov() function.

Value

Coefficient alpha

Examples

compute_alpha(sigma = reshape_vec2mat(cov = .4, order = 10))

compute_dmod

Comprehensive d_Mod calculator

Description

This is a general-purpose function to compute $d_{Mod}$ effect sizes from raw data and to perform bootstrapping. It subsumes the functionalities of the compute_dmod_par (for parametric computations) and compute_dmod_npar (for non-parametric computations) functions and automates the generation of regression equations and descriptive statistics for computing $d_{Mod}$ effect sizes. Please see documentation for compute_dmod_par and compute_dmod_npar for details about how the effect sizes are computed.
compute_dmod

Usage

compute_dmod(
  data,
  group,
  predictors,
  criterion,
  referent_id,
  focal_id_vec = NULL,
  conf_level = 0.95,
  rescale_cdf = TRUE,
  parametric = TRUE,
  bootstrap = TRUE,
  boot_iter = 1000,
  stratify = FALSE,
  empirical_ci = FALSE,
  cross_validate_wts = FALSE
)

Arguments

data Data frame containing the data to be analyzed (if not a data frame, must be an object convertible to a data frame via the as.data.frame function). The data set must contain a criterion variable, at least one predictor variable, and a categorical variable that identifies the group to which each case (i.e., row) in the data set belongs.

group Name or column-index number of the variable that identifies group membership in the data set.

predictors Name(s) or column-index number(s) of the predictor variable(s) in the data set. No predictor can be a factor-type variable. If multiple predictors are specified, they will be combined into a regression-weighted composite that will be carried forward to compute $d_{Mod}$ effect sizes.

- Note: If weights other than regression weights should be used to combine the predictors into a composite, the user must manually compute such a composite, include the composite in the data set, and identify the composite variable in predictors.

criterion Name or column-index number of the criterion variable in the data set. The criterion cannot be a factor-type variable.

referent_id Label used to identify the referent group in the group variable.

focal_id_vec Label(s) to identify the focal group(s) in the group variable. If NULL (the default), the specified referent group will be compared to all other groups.

conf_level Confidence level (between 0 and 1) to be used in generating confidence intervals. Default is .95

rescale_cdf Logical argument that indicates whether parametric $d_{Mod}$ results should be rescaled to account for using a cumulative density < 1 in the computations (TRUE; default) or not (FALSE).
compute_dmod

**parametric** Logical argument that indicates whether $d_{Mod}$ should be computed using an assumed normal distribution (TRUE; default) or observed frequencies (FALSE).

**bootstrap** Logical argument that indicates whether $d_{Mod}$ should be bootstrapped (TRUE; default) or not (FALSE).

**boot_iter** Number of bootstrap iterations to compute (default = 1000).

**stratify** Logical argument that indicates whether the random bootstrap sampling should be stratified (TRUE) or unstratified (FALSE; default).

**empirical_ci** Logical argument that indicates whether the bootstrapped confidence intervals should be computed from the observed empirical distributions (TRUE) or computed using bootstrapped means and standard errors via the normal-theory approach (FALSE).

**cross_validate_wts** Only relevant when multiple predictors are specified and bootstrapping is performed. Logical argument that indicates whether regression weights derived from the full sample should be used to combine predictors in the bootstrapped samples (TRUE) or if a new set of weights should be derived during each iteration of the bootstrapping procedure (FALSE; default).

**Value**

If bootstrapping is selected, the list will include:

- **point_estimate** A matrix of effect sizes ($d_{Mod_{Signed}}$, $d_{Mod_{Unsigned}}$, $d_{Mod_{Under}}$, $d_{Mod_{Over}}$), proportions of under- and over-predicted criterion scores, minimum and maximum differences, and the scores associated with minimum and maximum differences. All of these values are computed using the full data set.

- **bootstrap_mean** A matrix of the same statistics as the **point_estimate** matrix, but the values in this matrix are the means of the results from bootstrapped samples.

- **bootstrap_se** A matrix of the same statistics as the **point_estimate** matrix, but the values in this matrix are bootstrapped standard errors (i.e., the standard deviations of the results from bootstrapped samples).

- **bootstrap_CI_Lo** A matrix of the same statistics as the **point_estimate** matrix, but the values in this matrix are the lower confidence bounds of the results from bootstrapped samples.

- **bootstrap_CI_Hi** A matrix of the same statistics as the **point_estimate** matrix, but the values in this matrix are the upper confidence bounds of the results from bootstrapped samples.

If no bootstrapping is performed, the output will be limited to the **point_estimate** matrix.

**References**


**Examples**

```r
# Generate some hypothetical data for a referent group and three focal groups:
set.seed(10)
```
refDat <- MASS::mvrnorm(n = 1000, mu = c(.5, .2),
  Sigma = matrix(c(1, .5, .5, 1), 2, 2), empirical = TRUE)
foc1Dat <- MASS::mvrnorm(n = 1000, mu = c(-.5, -.2),
  Sigma = matrix(c(1, .5, .5, 1), 2, 2), empirical = TRUE)
foc2Dat <- MASS::mvrnorm(n = 1000, mu = c(0, 0),
  Sigma = matrix(c(1, .3, .3, 1), 2, 2), empirical = TRUE)
foc3Dat <- MASS::mvrnorm(n = 1000, mu = c(-.5, -.2),
  Sigma = matrix(c(1, .3, .3, 1), 2, 2), empirical = TRUE)
colnames(refDat) <- colnames(foc1Dat) <- colnames(foc2Dat) <- colnames(foc3Dat) <- c("X", "Y")
dat <- rbind(cbind(G = 1, refDat), cbind(G = 2, foc1Dat),
  cbind(G = 3, foc2Dat), cbind(G = 4, foc3Dat))

# Compute point estimates of parametric d_mod effect sizes:
compute_dmod(data = dat, group = "G", predictors = "X", criterion = "Y",
  referent_id = 1, focal_id_vec = 2:4,
  conf_level = .95, rescale_cdf = TRUE, parametric = TRUE,
  bootstrap = FALSE)

# Compute point estimates of non-parametric d_mod effect sizes:
compute_dmod(data = dat, group = "G", predictors = "X", criterion = "Y",
  referent_id = 1, focal_id_vec = 2:4,
  conf_level = .95, rescale_cdf = TRUE, parametric = FALSE,
  bootstrap = FALSE)

# Compute unstratified bootstrapped estimates of parametric d_mod effect sizes:
compute_dmod(data = dat, group = "G", predictors = "X", criterion = "Y",
  referent_id = 1, focal_id_vec = 2:4,
  conf_level = .95, rescale_cdf = TRUE, parametric = TRUE,
  boot_iter = 10, bootstrap = TRUE, stratify = FALSE, empirical_ci = FALSE)

# Compute unstratified bootstrapped estimates of non-parametric d_mod effect sizes:
compute_dmod(data = dat, group = "G", predictors = "X", criterion = "Y",
  referent_id = 1, focal_id_vec = 2:4,
  conf_level = .95, rescale_cdf = TRUE, parametric = FALSE,
  boot_iter = 10, bootstrap = TRUE, stratify = FALSE, empirical_ci = FALSE)

compute_dmod_npar  Function for computing non-parametric \( d_{\text{Mod}} \) effect sizes for a single focal group

Description

This function computes non-parametric \( d_{\text{Mod}} \) effect sizes from user-defined descriptive statistics and regression coefficients, using a distribution of observed scores as weights. This non-parametric function is best used when the assumption of normally distributed predictor scores is not reasonable and/or the distribution of scores observed in a sample is likely to represent the distribution of scores in the population of interest. If one has access to the full raw data set, the \( d_{\text{Mod}} \) function may be used as a wrapper to this function so that the regression equations and descriptive statistics can be computed automatically within the program.
Usage

\[
\text{compute_dmod_npar}( \\
\text{referent_int,} \\
\text{referent_slope,} \\
\text{focal_int,} \\
\text{focal_slope,} \\
\text{focal_x,} \\
\text{referent_sd_y})
\]

Arguments

- `referent_int`: Referent group’s intercept.
- `referent_slope`: Referent group’s slope.
- `focal_int`: Focal group’s intercept.
- `focal_slope`: Focal group’s slope.
- `focal_x`: Focal group’s vector of predictor scores.
- `referent_sd_y`: Referent group’s criterion standard deviation.

Details

The \(d_{\text{ModSigned}}\) effect size (i.e., the average of differences in prediction over the range of predictor scores) is computed as

\[
d_{\text{ModSigned}} = \frac{\sum_{i=1}^{m} n_i \left[X_i (b_{11} - b_{12}) + b_{01} - b_{02}\right]}{SD_{Y_1} \sum_{i=1}^{m} n_i},
\]

where

- \(SD_{Y_1}\) is the referent group’s criterion standard deviation;
- \(m\) is the number of unique scores in the distribution of focal-group predictor scores;
- \(X\) is the vector of unique focal-group predictor scores, indexed \(i = 1\) through \(m\);
- \(X_i\) is the \(i^{th}\) unique score value;
- \(n\) is the vector of frequencies associated with the elements of \(X\);
- \(n_i\) is the number of cases with a score equal to \(X_i\);
- \(b_{11}\) and \(b_{12}\) are the slopes of the regression of \(Y\) on \(X\) for the referent and focal groups, respectively; and
- \(b_{01}\) and \(b_{02}\) are the intercepts of the regression of \(Y\) on \(X\) for the referent and focal groups, respectively.

The \(d_{\text{ModUnder}}\) and \(d_{\text{ModOver}}\) effect sizes are computed using the same equation as \(d_{\text{ModSigned}}\), but \(d_{\text{ModUnder}}\) is the weighted average of all scores in the area of underprediction (i.e., the differences in prediction with negative signs) and \(d_{\text{ModOver}}\) is the weighted average of all scores in the area of overprediction (i.e., the differences in prediction with negative signs).
The \( d_{\text{Mod}_{\text{unsigned}}} \) effect size (i.e., the average of absolute differences in prediction over the range of predictor scores) is computed as

\[
d_{\text{Mod}_{\text{unsigned}}} = \frac{\sum_{i=1}^{m} n_i |X_i (b_{1i} - b_{12}) + b_{01} - b_{02}|}{SD_Y \sum_{i=1}^{m} n_i}.
\]

The \( d_{\text{Min}} \) effect size (i.e., the smallest absolute difference in prediction observed over the range of predictor scores) is computed as

\[
d_{\text{Min}} = \frac{1}{SD_Y} \text{Min} [|X (b_{11} - b_{12}) + b_{01} - b_{02}|].
\]

The \( d_{\text{Max}} \) effect size (i.e., the largest absolute difference in prediction observed over the range of predictor scores) is computed as

\[
d_{\text{Max}} = \frac{1}{SD_Y} \text{Max} [|X (b_{11} - b_{12}) + b_{01} - b_{02}|].
\]

**Note:** When \( d_{\text{Min}} \) and \( d_{\text{Max}} \) are computed in this package, the output will display the signs of the differences (rather than the absolute values of the differences) to aid in interpretation.

### Value

A vector of effect sizes (\( d_{\text{Mod}_{\text{Signed}}} \), \( d_{\text{Mod}_{\text{unsigned}}} \), \( d_{\text{Mod}_{\text{under}}} \), \( d_{\text{Mod}_{\text{over}}} \)), proportions of under- and over-predicted criterion scores, minimum and maximum differences (i.e., \( d_{\text{Mod}_{\text{under}}} \) and \( d_{\text{Mod}_{\text{over}}} \)), and the scores associated with minimum and maximum differences.

### Examples

```r
# Generate some hypothetical data for a referent group and three focal groups:
set.seed(10)
refDat <- MASS::mvrnorm(n = 1000, mu = c(.5, .2),
                        Sigma = matrix(c(1, .5, .5, 1), 2, 2), empirical = TRUE)
foc1Dat <- MASS::mvrnorm(n = 1000, mu = c(-.5, -.2),
                         Sigma = matrix(c(1, .5, .5, 1), 2, 2), empirical = TRUE)
foc2Dat <- MASS::mvrnorm(n = 1000, mu = c(0, 0),
                         Sigma = matrix(c(1, .3, .3, 1), 2, 2), empirical = TRUE)
foc3Dat <- MASS::mvrnorm(n = 1000, mu = c(-.5, -.2),
                         Sigma = matrix(c(1, .3, .3, 1), 2, 2), empirical = TRUE)
colnames(refDat) <- colnames(foc1Dat) <- colnames(foc2Dat) <- colnames(foc3Dat) <- c("X", "Y")

# Compute a regression model for each group:
refRegMod <- lm(Y ~ X, data.frame(refDat))$coef
foc1RegMod <- lm(Y ~ X, data.frame(foc1Dat))$coef
foc2RegMod <- lm(Y ~ X, data.frame(foc2Dat))$coef
foc3RegMod <- lm(Y ~ X, data.frame(foc3Dat))$coef

# Use the subgroup regression models to compute d_mod for each referent-focal pairing:
# Focal group #1:
compute_dmod_npar(referent_int = refRegMod[1], referent_slope = refRegMod[2],
                   focal_int = foc1RegMod[1], focal_slope = foc1RegMod[2],
                   ...)
```

compute_dmod_par

function for computing parametric d_Mod effect sizes for any number of focal groups

Description
This function computes $d_{Mod}$ effect sizes from user-defined descriptive statistics and regression coefficients. If one has access to a raw data set, the dMod function may be used as a wrapper to this function so that the regression equations and descriptive statistics can be computed automatically within the program.

Usage
compute_dmod_par(
  referent_int,
  referent_slope,
  focal_int,
  focal_slope,
  focal_mean_x,
  focal_sd_x,
  referent_sd_y,
  focal_min_x,
  focal_max_x,
  focal_names = NULL,
  rescale_cdf = TRUE
)

Arguments
referent_int Referent group’s intercept.
referent_slope Referent group’s slope.
focal_int Focal groups’ intercepts.
focal_slope Focal groups’ slopes.
focal_mean_x Focal groups’ predictor-score means.
focal_sd_x Focal groups’ predictor-score standard deviations.
referent_sd_y  Referent group’s criterion standard deviation.
focal_min_x  Focal groups’ minimum predictor scores.
focal_max_x  Focal groups’ maximum predictor scores.
focal_names  Focal-group names. If NULL (the default), the focal groups will be given numeric labels ranging from 1 through the number of groups.
rescale_cdf  Logical argument that indicates whether parametric \(d_{Mod}\) results should be rescaled to account for using a cumulative density < 1 in the computations (TRUE; default) or not (FALSE).

Details

The \(d_{Mod_{Signed}}\) effect size (i.e., the average of differences in prediction over the range of predictor scores) is computed as

\[
d_{Mod_{Signed}} = \frac{1}{SD_{Y_1}} \int f_2(X) \left[ X (b_{1_1} - b_{1_2}) + b_{0_1} - b_{0_2} \right] dX,
\]

where

- \(SD_{Y_1}\) is the referent group’s criterion standard deviation;
- \(f_2(X)\) is the normal-density function for the distribution of focal-group predictor scores;
- \(b_{1_1}\) and \(b_{1_2}\) are the slopes of the regression of \(Y\) on \(X\) for the referent and focal groups, respectively;
- \(b_{0_1}\) and \(b_{0_2}\) are the intercepts of the regression of \(Y\) on \(X\) for the referent and focal groups, respectively; and
- the integral spans all \(X\) scores within the operational range of predictor scores for the focal group.

The \(d_{Mod_{Under}}\) and \(d_{Mod_{Over}}\) effect sizes are computed using the same equation as \(d_{Mod_{Signed}}\), but \(d_{Mod_{Under}}\) is the weighted average of all scores in the area of underprediction (i.e., the differences in prediction with negative signs) and \(d_{Mod_{Over}}\) is the weighted average of all scores in the area of overprediction (i.e., the differences in prediction with negative signs).

The \(d_{Mod_{Unsigned}}\) effect size (i.e., the average of absolute differences in prediction over the range of predictor scores) is computed as

\[
d_{Mod_{Unsigned}} = \frac{1}{SD_{Y_1}} \int f_2(X) \left| X (b_{1_1} - b_{1_2}) + b_{0_1} - b_{0_2} \right| dX.
\]

The \(d_{Min}\) effect size (i.e., the smallest absolute difference in prediction observed over the range of predictor scores) is computed as

\[
d_{Min} = \frac{1}{SD_{Y_1}} Min \left[ \left| X (b_{1_1} - b_{1_2}) + b_{0_1} - b_{0_2} \right| \right].
\]

The \(d_{Max}\) effect size (i.e., the largest absolute difference in prediction observed over the range of predictor scores) is computed as

\[
d_{Max} = \frac{1}{SD_{Y_1}} Max \left[ \left| X (b_{1_1} - b_{1_2}) + b_{0_1} - b_{0_2} \right| \right].
\]
Note: When \( d_{\text{Min}} \) and \( d_{\text{Max}} \) are computed in this package, the output will display the signs of the differences (rather than the absolute values of the differences) to aid in interpretation.

If \( d_{\text{Mod}} \) effect sizes are to be rescaled to compensate for a cumulative density less than 1 (see the rescale_cdf argument), the result of each effect size involving integration will be divided by the ratio of the cumulative density of the observed range of scores (i.e., the range bounded by the focal_min_x and focal_max_x arguments) to the cumulative density of scores bounded by \(-\text{Inf}\) and \(\text{Inf}\).

Value

A matrix of effect sizes (\( d_{\text{ModSigned}}, d_{\text{ModUnsigned}}, d_{\text{ModUnder}}, d_{\text{ModOver}} \)), proportions of under- and over-predicted criterion scores, minimum and maximum differences (i.e., \( d_{\text{ModUnder}} \) and \( d_{\text{ModOver}} \)), and the scores associated with minimum and maximum differences. Note that if the regression lines are parallel and infinite focal_min_x and focal_max_x values were specified, the extrema will be defined using the scores 3 focal-group SDs above and below the corresponding focal-group means.

References


Examples

```r
compute_dmod_par(referent_int = -.05, referent_slope = .5,
  focal_int = c(.05, 0, -.05), focal_slope = c(.5, .3, .3),
  focal_mean_x = c(-.5, 0, -.5), focal_sd_x = rep(1, 3),
  referent_sd_y = 1,
  focal_min_x = rep(-Inf, 3), focal_max_x = rep(Inf, 3),
  focal_names = NULL, rescale_cdf = TRUE)
```

Description

Confidence limits for noncentral chi square parameters (function and documentation from package 'MBESS' version 4.4.3) Function to determine the noncentral parameter that leads to the observed Chi.Square-value, so that a confidence interval for the population noncentral chi-square value can be formed.
conf.limits.nc.chisq

Usage

conf.limits.nc.chisq(
  Chi.Square = NULL,
  conf.level = 0.95,
  df = NULL,
  alpha.lower = NULL,
  alpha.upper = NULL,
  tol = 1e-09,
  Jumping.Prop = 0.1
)

Arguments

Chi.Square  the observed chi-square value
conf.level  the desired degree of confidence for the interval
df          the degrees of freedom
alpha.lower Type I error for the lower confidence limit
alpha.upper Type I error for the upper confidence limit
tol         tolerance for iterative convergence
Jumping.Prop Value used in the iterative scheme to determine the noncentral parameters necessary for confidence interval construction using noncentral chi square-distributions (0 < Jumping.Prop < 1)

Details

If the function fails (or if a function relying upon this function fails), adjust the Jumping.Prop (to a smaller value).

Value

• Lower.Limit Value of the distribution with Lower.Limit noncentral value that has at its specified quantile Chi.Square
• Prob.Less.Lower Proportion of cases falling below Lower.Limit
• Upper.Limit Value of the distribution with Upper.Limit noncentral value that has at its specified quantile Chi.Square
• Prob.Greater.Upper Proportion of cases falling above Upper.Limit

Author(s)

Ken Kelley (University of Notre Dame; <KKelley@ND.edu>), Keke Lai (University of California–Merced)
confidence

Construct a confidence interval

Description

Function to construct a confidence interval around an effect size or mean effect size.

Usage

confidence(
  mean,
  se = NULL,
  df = NULL,
  conf_level = 0.95,
  conf_method = c("t", "norm"),
  ...
)

Arguments

mean Mean effect size (if used in a meta-analysis) or observed effect size (if used on individual statistics).
se Standard error of the statistic.
df Degrees of freedom of the statistic (necessary if using the \( t \) distribution).
conf_level Confidence level that defines the width of the confidence interval (default = .95).
conf_method Distribution to be used to compute the width of confidence intervals. Available options are "t" for \( t \) distribution or "norm" for normal distribution.
...

Details

\[
CI = mean_{es} \pm \text{quantile} \times SE_{es}
\]

Value

A matrix of confidence intervals of the specified width.

Examples

confidence(mean = c(.3, .5), se = c(.15, .2), df = c(100, 200), conf_level = .95, conf_method = "t")
confidence(mean = c(.3, .5), se = c(.15, .2), conf_level = .95, conf_method = "norm")
confidence_r

Construct a confidence interval for correlations using Fisher's z transformation

Description

Construct a confidence interval for correlations using Fisher's z transformation

Usage

confidence_r(r, n, conf_level = 0.95)

Arguments

- r: A vector of correlations
- n: A vector of sample sizes
- conf_level: Confidence level that defines the width of the confidence interval (default = .95).

Value

A confidence interval of the specified width (or matrix of confidence intervals)

Examples

confidence_r(r = .3, n = 200, conf_level = .95)

confint

Confidence interval method for objects of classes deriving from "lm_mat"

Description

Confidence interval method for objects of classes deriving from "lm_mat." Returns lower and upper bounds of confidence intervals for regression coefficients.

Arguments

- object: Matrix regression object.
- parm: a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered.
- level: Confidence level
- ... further arguments passed to or from other methods.
control_intercor

Control function to curate intercorrelations to be used in automatic compositing routine

Description

Control function to curate intercorrelations to be used in automatic compositing routine

Usage

control_intercor(
  rxyi = NULL,
  n = NULL,
  sample_id = NULL,
  construct_x = NULL,
  construct_y = NULL,
  construct_names = NULL,
  facet_x = NULL,
  facet_y = NULL,
  intercor_vec = NULL,
  intercor_scalar = 0.5,
  dx = NULL,
  dy = NULL,
  p = 0.5,
  partial_intercor = FALSE,
  data = NULL,
  ...
)

Arguments

rxyi Vector or column name of observed correlations.
n Vector or column name of sample sizes.
sample_id Vector of identification labels for samples/studies in the meta-analysis.
construct_x, construct_y Vector of construct names for constructs designated as "X" or "Y".
construct_names Vector of all construct names to be included in the meta-analysis.
facet_x, facet_y Vector of facet names for constructs designated as "X" or "Y".
intercor_vec Named vector of pre-specified intercorrelations among measures of constructs in the meta-analysis.
intercor_scalar Generic scalar intercorrelation that can stand in for unobserved or unspecified values.
dx, dy  

\(d\) values corresponding to `construct_x` and `construct_y`. These values only need to be supplied for cases in which \(r_{xyi}\) represents a correlation between two measures of the same construct.

p  

Scalar or vector containing the proportions of group membership corresponding to the \(d\) values.

`partial_intercor`  

For meta-analyses of \(d\) values only: Logical scalar, vector, or column corresponding to values in \(r_{xyi}\) that determines whether the correlations are to be treated as within-group correlations (i.e., partial correlation controlling for group membership; \texttt{TRUE}) or not (\texttt{FALSE}; default). Note that this only converts correlation values from the \(r_{yi}\) argument - any values provided in the `intercor_vec` or `intercor_scalar` arguments must be total correlations or converted to total correlations using the `mix_r_2group()` function prior to running `control_intercor`.

data  

Data frame containing columns whose names may be provided as arguments to vector arguments.

...  

Further arguments to be passed to functions called within the meta-analysis.

Value  

A vector of intercorrelations

Examples

```r
## Create a dataset in which constructs correlate with themselves
rxyi <- seq(.1, .5, length.out = 27)
construct_x <- rep(rep(c("X", "Y", "Z"), 3), 3)
construct_y <- c(rep("X", 9), rep("Y", 9), rep("Z", 9))
dat <- data.frame(rxyi = rxyi,
                   construct_x = construct_x,
                   construct_y = construct_y,
                   stringsAsFactors = FALSE)
dat <- rbind(cbind(sample_id = "Sample 1", dat),
             cbind(sample_id = "Sample 2", dat),
             cbind(sample_id = "Sample 3", dat))

## Identify some constructs for which intercorrelations are not
## represented in the data object:
construct_names = c("U", "V", "W")

## Specify some externally determined intercorrelations among measures:
intercor_vec <- c(W = .4, X = .1)

## Specify a generic scalar intercorrelation that can stand in for missing values:
intercor_scalar <- .5

control_intercor(rxyi = rxyi, sample_id = sample_id,
                  construct_x = construct_x, construct_y = construct_y,
                  construct_names = construct_names,
                  intercor_vec = intercor_vec, intercor_scalar = intercor_scalar, data = dat)
```
**control_psychmeta**

Control for *psychmeta* meta-analyses

### Description

Control for *psychmeta* meta-analyses

### Usage

```r
control_psychmeta(
  error_type = c("mean", "sample"),
  conf_level = 0.95,
  cred_level = 0.8,
  conf_method = c("t", "norm"),
  cred_method = c("t", "norm"),
  var_unbiased = TRUE,
  pairwise_ads = FALSE,
  moderated_ads = FALSE,
  residual_ads = TRUE,
  check_dependence = TRUE,
  collapse_method = c("composite", "average", "stop"),
  intercor = control_intercor(),
  clean_artifacts = TRUE,
  impute_artifacts = TRUE,
  impute_method = c("bootstrap_mod", "bootstrap_full", "simulate_mod", "simulate_full",
                   "wt_mean_mod", "wt_mean_full", "unwt_mean_mod", "unwt_mean_full", "replace Unity",
                   "stop"),
  seed = 42,
  use_all_arts = TRUE,
  estimate_pa = FALSE,
  decimals = 2,
  hs_override = FALSE,
  zero_substitute = .Machine$double.eps,
  ...
)
```

### Arguments

- **error_type**
  Method to be used to estimate error variances: "mean" uses the mean effect size to estimate error variances and "sample" uses the sample-specific effect sizes.

- **conf_level**
  Confidence level to define the width of the confidence interval (default = .95).

- **cred_level**
  Credibility level to define the width of the credibility interval (default = .80).

- **conf_method**
  Distribution to be used to compute the width of confidence intervals. Available options are "t" for t distribution or "norm" for normal distribution.

- **cred_method**
  Distribution to be used to compute the width of credibility intervals. Available options are "t" for t distribution or "norm" for normal distribution.
var_unbiased Logical scalar determining whether variances should be unbiased (TRUE) or maximum-likelihood (FALSE).

pairwise_ads Logical value that determines whether to compute artifact distributions in a construct-pair-wise fashion (TRUE) or separately by construct (FALSE, default).

moderated_ads Logical value that determines whether to compute artifact distributions separately for each moderator combination (TRUE) or for overall analyses only (FALSE, default).

residual_ads Logical argument that determines whether to use residualized variances (TRUE) or observed variances (FALSE) of artifact distributions to estimate sd_rho.

check_dependence Logical scalar that determines whether database should be checked for violations of independence (TRUE) or not (FALSE).

collapse_method Character argument that determines how to collapse dependent studies. Options are "composite" (default), "average," and "stop."

intercor The intercorrelation(s) among variables to be combined into a composite. Can be a scalar, a named vector with element named according to the names of constructs, or output from the control_intercor function. Default scalar value is .5.

clean_artifacts If TRUE, multiple instances of the same construct (or construct-measure pair, if measure is provided) in the database are compared and reconciled with each other in the case that any of the matching entries within a study have different artifact values. When impute_method is anything other than "stop", this method is always implemented to prevent discrepancies among imputed values.

impute_artifacts If TRUE, artifact imputation will be performed (see impute_method for imputation procedures). Default is FALSE for artifact-distribution meta-analyses and TRUE otherwise. When imputation is performed, clean_artifacts is treated as TRUE so as to resolve all discrepancies among artifact entries before and after imputation.

impute_method Method to use for imputing artifacts. Choices are:

- bootstrap_mod
  Select random values from the most specific moderator categories available (default).
- bootstrap_full
  Select random values from the full vector of artifacts.
- simulate_mod
  Generate random values from the distribution with the mean and variance of observed artifacts from the most specific moderator categories available. (uses rnorm for u ratios and rbeta for reliability values).
- simulate_full
  Generate random values from the distribution with the mean and variance of all observed artifacts (uses rnorm for u ratios and rbeta for reliability values).
• wt_mean_mod
Replace missing values with the sample-size weighted mean of the distribution of artifacts from the most specific moderator categories available (not recommended).

• wt_mean_full
Replace missing values with the sample-size weighted mean of the full distribution of artifacts (not recommended).

• unwt_mean_mod
Replace missing values with the unweighted mean of the distribution of artifacts from the most specific moderator categories available (not recommended).

• unwt_mean_full
Replace missing values with the unweighted mean of the full distribution of artifacts (not recommended).

• replace_unity
Replace missing values with 1 (not recommended).

• stop
Stop evaluations when missing artifacts are encountered.

If an imputation method ending in "mod" is selected but no moderators are provided, the "mod" suffix will internally be replaced with "full".

seed
Seed value to use for imputing artifacts in a reproducible way. Default value is 42.

use_all_arts
Logical scalar that determines whether artifact values from studies without valid effect sizes should be used in artifact distributions (TRUE; default) or not (FALSE).

estimate_pa
Logical scalar that determines whether the unrestricted subgroup proportions associated with univariate-range-restricted effect sizes should be estimated by rescaling the range-restricted subgroup proportions as a function of the range-restriction correction (TRUE) or not (FALSE; default).

decimals
Number of decimal places to which interactive artifact distributions should be rounded (default is 2 decimal places).

hs_override
When TRUE, this will override settings for wt_type (will set to "sample_size"), error_type (will set to "mean"), correct_bias (will set to TRUE), conf_method (will set to "norm"), cred_method (will set to "norm"), var_unbiased (will set to FALSE), residual_ads (will be set to FALSE), and use_all_arts (will be set to FALSE).

zero SUBSTITUTE
Value to be used as a functionally equivalent substitute for exactly zero effect sizes in individual-correction meta-analyses to facilitate the estimation of corrected error variances. By default, this is set to .Machine$double.eps.

... Further arguments to be passed to functions called within the meta-analysis.

Value
A list of control arguments in the package environment.
Examples

control_psychmeta()

convert_es  \hspace{1cm} \textit{Convert effect sizes}

Description

This function converts a variety of effect sizes to correlations, Cohen’s \(d\) values, or common language effect sizes, and calculates sampling error variances and effective sample sizes.

Usage

\begin{verbatim}
convert_es(
\end{verbatim}

Arguments

- \textbf{es} Vector of effect sizes to convert.
- \textbf{input_es} Scalar. Metric of input effect sizes. Currently supports correlations, Cohen’s \(d\), independent samples \(t\) values (or their \(p\) values), two-group one-way ANOVA \(F\) values (or their \(p\) values), 1-df \(\chi^2\) values (or their \(p\) values), odds ratios, log odds ratios, Fisher \(z\), and the common language effect size (CLES, \(A\), AUC).
- \textbf{output_es} Scalar. Metric of output effect sizes. Currently supports correlations, Cohen’s \(d\) values, and common language effect sizes (CLES, \(A\), AUC).
- \textbf{n1} Vector of total sample sizes or sample sizes of group 1 of the two groups being contrasted.
- \textbf{n2} Vector of sample sizes of group 2 of the two groups being contrasted.
- \textbf{df1} Vector of input test statistic degrees of freedom (for \(t\) and \(\chi^2\)) or between-groups degree of freedom (for \(F\)).
- \textbf{df2} Vector of input test statistic within-group degrees of freedom (for \(F\)).
- \textbf{sd1} Vector of pooled (within-group) standard deviations or standard deviations of group 1 of the two groups being contrasted.
- \textbf{sd2} Vector of standard deviations of group 2 of the two groups being contrasted.
- \textbf{tails} Vector of the tails for \(p\) values when \texttt{input_es} = "p.t". Can be 2 (default) or 1.
Value

A data frame of class es with variables:

- \( r, d, A \)  The converted effect sizes
- \( n_{\text{effective}} \)  The effective total sample size
- \( n \)  The total number of cases (original sample size)
- \( n_1, n_2 \)  If applicable, subgroup sample sizes
- \( \text{var}_e \)  The estimated sampling error variance

References


Examples

```r
convert_es(es = 1, input_es="d", output_es="r", n1=100)
convert_es(es = 1, input_es="d", output_es="r", n1=50, n2 = 50)
convert_es(es = .2, input_es="r", output_es="d", n1=100, n2=150)
convert_es(es = -1.3, input_es="t", output_es="r", n1 = 100, n2 = 140)
convert_es(es = 10.3, input_es="F", output_es="d", n1 = 100, n2 = 150)
convert_es(es = 1.3, input_es="chisq", output_es="r", n1 = 100, n2 = 100)
convert_es(es = .021, input_es="p.chisq", output_es="d", n1 = 100, n2 = 100)
convert_es(es = 4.37, input_es="or", output_es="r", n1=100, n2=100)
convert_es(es = 4.37, input_es="or", output_es="d", n1=100, n2=100)
convert_es(es = 1.47, input_es="lor", output_es="r", n1=100, n2=100)
convert_es(es = 1.47, input_es="lor", output_es="d", n1=100, n2=100)
```

---

**convert_ma**  
*Function to convert meta-analysis of correlations to d values or vice-versa*

**Description**

Takes a meta-analysis class object of \( d \) values or correlations (classes \( r\_as\_r, d\_as\_d, r\_as\_d, \) and \( d\_as\_r \); second-order meta-analyses are currently not supported) as an input and uses conversion formulas and Taylor series approximations to convert effect sizes and variance estimates, respectively.
Usage

```r
convert_ma(ma_obj, ...)
convert_meta(ma_obj, ...)
```

Arguments

- **ma_obj**: A meta-analysis object of class `r_as_r`, `d_as_d`, `r_as_d`, or `d_as_r`
- `...`: Additional arguments.

Details

The formula used to convert correlations to `d` values is:

\[
d = r \sqrt{\frac{1}{p(1-p)}} \sqrt{1-r^2}
\]

The formula used to convert `d` values to correlations is:

\[
r = \frac{d}{\sqrt{d^2 + \frac{1}{p(1-p)}}}
\]

To approximate the variance of correlations from the variance of `d` values, the function computes:

\[
\text{var}_r \approx a_d^2 \text{var}_d
\]

where \(a_d\) is the first partial derivative of the `d`-to-`r` transformation with respect to `d`:

\[
a_d = -\frac{1}{[d^2p(1-p) - 1] \sqrt{d^2 + \frac{1}{p-p^2}}}
\]

To approximate the variance of `d` values from the variance of correlations, the function computes:

\[
\text{var}_d \approx a_r^2 \text{var}_r
\]

where \(a_r\) is the first partial derivative of the `r`-to-`d` transformation with respect to `r`:

\[
a_r = \frac{\sqrt{\frac{1}{p-p^2}}}{(1-r^2)^{1.5}}
\]

Value

A meta-analysis converted to the `d` value metric (if `ma_obj` was a meta-analysis in the correlation metric) or converted to the correlation metric (if `ma_obj` was a meta-analysis in the `d` value metric).
correct_d

Correct d values for measurement error and/or range restriction

Description

This function is a wrapper for the `correct_r()` function to correct d values for statistical and psychometric artifacts.

Usage

```r
correct_d(
correction = c("meas", "uvdrr_g", "uvdrr_y", "uvirr_g", "uvirr_y", "bvdrr", "bvirr"),
d,
ryy = 1,
uy = 1,
rgg = 1,
pi = NULL,
apa = NULL,
uy.observed = TRUE,
ryy.restricted = TRUE,
ryy.type = "alpha",
k.items_y = NA,
sign rgz = 1,
sign ryz = 1,
n1 = NULL,
n2 = NA,
conf.level = 0.95,
correct.bias = FALSE
)
```

Arguments

- `correction`: Type of correction to be applied. Options are "meas", "uvdrr_g", "uvdrr_y", "uvirr_g", "uvirr_y", "bvdrr", "bvirr"
- `d`: Vector of d values.
- `ryy`: Vector of reliability coefficients for Y (the continuous variable).
- `uy`: Vector of u ratios for Y (the continuous variable).
- `rgg`: Vector of reliabilities for the group variable (i.e., the correlations between observed group membership and latent group membership).
- `pi`: Proportion of cases in one of the groups in the observed data (not necessary if `n1` and `n2` reflect this proportionality).
- `apa`: Proportion of cases in one of the groups in the population.
- `uy.observed`: Logical vector in which each entry specifies whether the corresponding uy value is an observed-score u ratio (TRUE) or a true-score u ratio. All entries are TRUE by default.
**rry_restricted** Logical vector in which each entry specifies whether the corresponding rxx value is an incumbent reliability (TRUE) or an applicant reliability. All entries are TRUE by default.

**rry_type** String vector identifying the types of reliability estimates supplied (e.g., "alpha", "retest", "interrater_r", "splithalf"). See the documentation for ma_r() for a full list of acceptable reliability types.

**k_items_y** Numeric vector identifying the number of items in each scale.

**sign_ryz** Vector of signs of the relationships between Y variables and the selection mechanism.

**sign_rgz** Vector of signs of the relationships between grouping variables and the selection mechanism.

**n1** Optional vector of sample sizes associated with group 1 (or the total sample size, if n2 is NULL).

**n2** Optional vector of sample sizes associated with group 2.

**conf_level** Confidence level to define the width of the confidence interval (default = .95).

**correct_bias** Logical argument that determines whether to correct error-variance estimates for small-sample bias in correlations (TRUE) or not (FALSE). For sporadic corrections (e.g., in mixed artifact-distribution meta-analyses), this should be set to FALSE (the default).

**Value**

Data frame(s) of observed $d$ values ($dgyi$), operational range-restricted $d$ values corrected for measurement error in Y only ($dgpi$), operational range-restricted $d$ values corrected for measurement error in the grouping only ($dGyi$), range-restricted true-score $d$ values ($dGpi$), range-corrected observed-score $d$ values ($dgya$), operational range-corrected $d$ values corrected for measurement error in Y only ($dgpa$), operational range-corrected $d$ values corrected for measurement error in the grouping only ($dGya$), and range-corrected true-score $d$ values ($dGpa$).

**References**


Examples

```r
## Correction for measurement error only
correct_d(correction = "meas", d = .5, ryy = .8, uy = .7,
          rGg = .9, pi = .7, pa = .5)
correct_d(correction = "meas", d = .5, ryy = .8, uy = .7,
          rGg = .9, pi = NULL, pa = .5, n1 = 100, n2 = 200)

## Correction for direct range restriction in the continuous variable
correct_d(correction = "uvdrr_y", d = .5, ryy = .8, uy = .7,
          rGg = .9, pi = .7, pa = .5)
correct_d(correction = "uvdrr_y", d = .5, ryy = .8, uy = .7,
          rGg = .9, pi = NULL, pa = .5, n1 = 100, n2 = 200)

## Correction for direct range restriction in the grouping variable
correct_d(correction = "uvdrr_g", d = .5, ryy = .8, uy = .7,
          rGg = .9, pi = .7, pa = .5)
correct_d(correction = "uvdrr_g", d = .5, ryy = .8, uy = .7,
          rGg = .9, pi = NULL, pa = .5, n1 = 100, n2 = 200)

## Correction for indirect range restriction in the continuous variable
correct_d(correction = "uvdrr_y", d = .5, ryy = .8, uy = .7,
          rGg = .9, pi = .7, pa = .5)
correct_d(correction = "uvdrr_y", d = .5, ryy = .8, uy = .7,
          rGg = .9, pi = NULL, pa = .5, n1 = 100, n2 = 200)

## Correction for indirect range restriction in the grouping variable
correct_d(correction = "uvdrr_g", d = .5, ryy = .8, uy = .7,
          rGg = .9, pi = .7, pa = .5)
correct_d(correction = "uvdrr_g", d = .5, ryy = .8, uy = .7,
          rGg = .9, pi = NULL, pa = .5, n1 = 100, n2 = 200)

## Correction for direct range restriction in both variables
correct_d(correction = "bvdrr", d = .5, ryy = .8, uy = .7,
          rGg = .9, pi = .7, pa = .5)
correct_d(correction = "bvdrr", d = .5, ryy = .8, uy = .7,
          rGg = .9, pi = NULL, pa = .5, n1 = 100, n2 = 200)

## Correction for indirect range restriction in both variables
correct_d(correction = "bvirr", d = .5, ryy = .8, uy = .7,
          rGg = .9, pi = .7, pa = .5)
correct_d(correction = "bvirr", d = .5, ryy = .8, uy = .7,
          rGg = .9, pi = NULL, pa = .5, n1 = 100, n2 = 200)
```

**correct_d_bias**

Correct for small-sample bias in Cohen’s $d$ values.
**Description**

Corrects a vector of Cohen’s $d$ values for small-sample bias, as Cohen’s $d$ has a slight positive bias. The bias-corrected $d$ value is often called Hedges’s $g$.

**Usage**

```r
correct_d_bias(d, n)
```

**Arguments**

- `d` Vector of Cohen’s $d$ values.
- `n` Vector of sample sizes.

**Details**

The bias correction is:

$$g = d_c = d_{obs} \times J$$

where

$$J = \frac{\Gamma\left(\frac{n-2}{2}\right)}{\sqrt{\frac{n-2}{2} \times \Gamma\left(\frac{n-3}{2}\right)}}$$

and $d_{obs}$ is the observed effect size, $g = d_c$ is the corrected (unbiased) estimate, $n$ is the total sample size, and $\Gamma()$ is the [gamma function](https://en.wikipedia.org/wiki/Gamma_function).

Historically, using the gamma function was computationally intensive, so an approximation for $J$ was used (Borenstein et al., 2009):

$$J = 1 - 3/(4 \times (n - 2) - 1)$$

This approximation is no longer necessary with modern computers.

**Value**

Vector of $g$ values (d values corrected for small-sample bias).

**References**


**Examples**

```r
correct_d_bias(d = .3, n = 30)
correct_d_bias(d = .3, n = 300)
correct_d_bias(d = .3, n = 3000)
```
Correct for small-sample bias in Glass’ $\Delta$ values

**Description**

Correct for small-sample bias in Glass’ $\Delta$ values.

**Usage**

```r
correct_glass_bias(delta, nc, ne, use_pooled_sd = rep(FALSE, length(delta)))
```

**Arguments**

- `delta` Vector of Glass’ $\Delta$ values.
- `nc` Vector of control-group sample sizes.
- `ne` Vector of experimental-group sample sizes.
- `use_pooled_sd` Logical vector determining whether the pooled standard deviation was used (TRUE) or not (FALSE; default).

**Details**

The bias correction is estimated as:

$$
\Delta_c = \Delta_{obs} \frac{\Gamma \left( \frac{n_{control} - 1}{2} \right)}{\Gamma \left( \frac{n_{control} - 1}{2} \right) \Gamma \left( \frac{n_{control} - 2}{2} \right)}
$$

where $\Delta$ is the observed effect size, $\Delta_c$ is the corrected estimate of $\Delta$, $n_{control}$ is the control-group sample size, and $\Gamma()$ is the gamma function.

**Value**

Vector of $d$ values corrected for small-sample bias.

**References**


**Examples**

```r
correct_glass_bias(delta = .3, nc = 30, ne = 30)
```
correct_matrix_mvrr  
Multivariate select/correction for covariance matrices

Description

Correct (or select upon) a covariance matrix using the Pearson-Aitken-Lawley multivariate selection theorem.

Usage

```r
correct_matrix_mvrr(
  Sigma_i,  
  Sigma_xx_a,  
  x_col,  
  y_col = NULL,  
  standardize = FALSE,  
  var_names = NULL
)
```

Arguments

- **Sigma_i**: The complete range-restricted (unrestricted) covariance matrix to be corrected (selected upon).
- **Sigma_xx_a**: The matrix of unrestricted (range-restricted) covariances among of selection variables.
- **x_col**: The row/column indices of the variables in `Sigma_i` that correspond, in order, to the variables in `Sigma_xx_a`.
- **y_col**: Optional: The variables in `Sigma_i` not listed in `x_col` that are to be manipulated by the multivariate range-restriction formula.
- **standardize**: Should the function’s output matrix be returned in standardized form (TRUE) or in unstandardized form (FALSE; the default).
- **var_names**: Optional vector of names for the variables in `Sigma_i`, in order of appearance in the matrix.

Value

A matrix that has been manipulated by the multivariate range-restriction formula.

References


Examples

Sigma_i <- reshape_vec2mat(cov = .2, var = .8, order = 4)
Sigma_xx_a <- reshape_vec2mat(cov = .5, order = 2)
correct_matrix_mvrr(Sigma_i = Sigma_i, Sigma_xx_a = Sigma_xx_a, x_col = 1:2, standardize = TRUE)

correct_means_mvrr Multivariate select/correction for vectors of means

Description

Correct (or select upon) a vector of means using principles from the Pearson-Aitken-Lawley multivariate selection theorem.

Usage

correct_means_mvrr(
  Sigma,
  means_i = rep(0, ncol(Sigma)),
  means_x_a,
  x_col,
  y_col = NULL,
  var_names = NULL
)

Arguments

Sigma The complete covariance matrix to be used to manipulate means: This matrix may be entirely unrestricted or entirely range restricted, as the regression weights estimated from this matrix are expected to be invariant to the effects of selection.
means_i The complete range-restricted (unrestricted) vector of means to be corrected (selected upon).
means_x_a The vector of unrestricted (range-restricted) means of selection variables
x_col The row/column indices of the variables in Sigma that correspond, in order, to the variables in means_x_a
y_col Optional: The variables in Sigma not listed in x_col that are to be manipulated by the multivariate range-restriction formula.
var_names Optional vector of names for the variables in Sigma, in order of appearance in the matrix.

Value

A vector of means that has been manipulated by the multivariate range-restriction formula.
References


Examples

```r
Sigma <- diag(.5, 4)
Sigma[lower.tri(Sigma)] <- c(.2, .3, .4, .3, .4, .4)
Sigma <- Sigma + t(Sigma)
diag(Sigma) <- 1
correct_means_mvrr(Sigma = Sigma, means_i = c(.3, .3, .1, .1),
means_x_a = c(-.1, -.1), x_col = 1:2)
```

---

correct_r  
Correct correlations for range restriction and/or measurement error

Description

Corrects Pearson correlations (r) for range restriction and/or measurement error

Usage

```r
correct_r(
  correction = c("meas", "uvdrr_x", "uvdrr_y", "uvirr_x", "uvirr_y", "bvdrr", "bvirr"),
  rxyi,
  ux = 1,
  uy = 1,
  rxx = 1,
  ryy = 1,
  ux_observed = TRUE,
  uy_observed = TRUE,
  rxx_restricted = TRUE,
  rxx_type = "alpha",
  k_items_x = NA,
  ryy_restricted = TRUE,
  ryy_type = "alpha",
  k_items_y = NA,
  sign_rxz = 1,
  sign_ryz = 1,
  n = NULL,
  conf_level = 0.95,
  correct_bias = FALSE,
  zero_substitute = .Machine$double.eps
)
```
Arguments

correction Type of correction to be applied. Options are "meas", "uvdrr_x", "uvdrr_y", "uvirr_x", "uvirr_y", "bvdrr", "bvirr"

rxyi Vector of observed correlations. **NOTE**: Beginning in *psychmeta* version 2.5.2, rxyi values of exactly 0 in individual-correction meta-analyses are replaced with a functionally equivalent value via the zero_substitute argument to facilitate the estimation of effective sample sizes.

ux Vector of u ratios for X.

uy Vector of u ratios for Y.

rxx Vector of reliability coefficients for X.

ryy Vector of reliability coefficients for Y.

ux_observed Logical vector in which each entry specifies whether the corresponding ux value is an observed-score u ratio (TRUE) or a true-score u ratio. All entries are TRUE by default.

uy_observed Logical vector in which each entry specifies whether the corresponding uy value is an observed-score u ratio (TRUE) or a true-score u ratio. All entries are TRUE by default.

rxx_restricted Logical vector in which each entry specifies whether the corresponding rxx value is an incumbent reliability (TRUE) or an applicant reliability. All entries are TRUE by default.

rxx_type, ryy_type String vector identifying the types of reliability estimates supplied (e.g., "alpha", "retest", "interrater_r", "splithalf"). See the documentation for *ma_r* for a full list of acceptable reliability types.

k_items_x, k_items_y Numeric vector identifying the number of items in each scale.

ryy_restricted Logical vector in which each entry specifies whether the corresponding rxx value is an incumbent reliability (TRUE) or an applicant reliability. All entries are TRUE by default.

sign_rxz Vector of signs of the relationships between X variables and the selection mechanism.

sign_ryz Vector of signs of the relationships between Y variables and the selection mechanism.

n Optional vector of sample sizes associated with the rxyi correlations.

conf_level Confidence level to define the width of the confidence interval (default = .95).

correct_bias Logical argument that determines whether to correct error-variance estimates for small-sample bias in correlations (TRUE) or not (FALSE). For sporadic corrections (e.g., in mixed artifact-distribution meta-analyses), this should be set to FALSE, the default).

zero_substitute Value to be used as a functionally equivalent substitute for exactly zero effect sizes to facilitate the estimation of effective sample sizes. By default, this is set to .Machine$double.eps.
Details

The correction for measurement error is:

$$\rho_{TP} = \frac{\rho_{XY}}{\sqrt{\rho_{XX}\rho_{YY}}}$$

The correction for univariate direct range restriction is:

$$\rho_{TPu} = \frac{\rho_{XYi}}{u_X\sqrt{\rho_{YYi}}}\sqrt{\left(\frac{1}{u_X} - 1\right)\frac{\rho_{XYi}^2}{\rho_{YYi}} + 1} / \sqrt{\rho_{XXu}}$$

The correction for univariate indirect range restriction is:

$$\rho_{TPu} = \frac{\rho_{XYi}}{u_T\sqrt{\rho_{XXi}\rho_{YYi}}}\sqrt{\left(\frac{1}{u_T^2} - 1\right)\frac{\rho_{XYi}^2}{\rho_{XXi}\rho_{YYi}} + 1}$$

The correction for bivariate direct range restriction is:

$$\rho_{TPa} = \frac{\rho_{XY} - 1}{2\rho_{XYi}}u_Xu_Y + \text{sign} (\rho_{XYi}) \sqrt{\frac{(1-\rho_{XYi})^2}{4\rho_{XYi}}}u_X^2u_Y^2 + 1 \sqrt{\rho_{XXa}\rho_{YYa}}$$

The correction for bivariate indirect range restriction is:

$$\rho_{TPa} = \frac{\rho_{XY} u_X u_Y + \lambda \sqrt{|1-u_X^2||1-u_Y^2|}}{\sqrt{\rho_{XXa}\rho_{YYa}}}$$

where the $\lambda$ value allows $u_X$ and $u_Y$ to fall on either side of unity so as to function as a two-stage correction for mixed patterns of range restriction and range enhancement. The $\lambda$ value is computed as:

$$\lambda = \text{sign} [\rho_{STa}\rho_{SPa} (1-u_X) (1-u_Y)] \frac{\text{sign} (1-u_X) \min \left(u_X, \frac{1}{u_X}\right) + \text{sign} (1-u_Y) \min \left(u_Y, \frac{1}{u_Y}\right)}{\min \left(u_X, \frac{1}{u_X}\right) \min \left(u_Y, \frac{1}{u_Y}\right)}$$

Value

Data frame(s) of observed correlations ($r_{XYi}$), operational range-restricted correlations corrected for measurement error in Y only ($r_{Xpi}$), operational range-restricted correlations corrected for measurement error in X only ($r_{Yti}$), and range-restricted true-score correlations ($r_{tpi}$), range-corrected observed-score correlations corrected for measurement error in Y only ($r_{xya}$), operational range-corrected correlations corrected for measurement error in Y only ($r_{xpa}$), operational range-corrected correlations corrected for measurement error in X only ($r_{tya}$), and range-corrected true-score correlations ($r_{tpa}$).
References


Examples

```r
## Correction for measurement error only
correct_r(correction = "meas", rxyi = .3, rxx = .8, ryy = .8,
ux_observed = TRUE, uy_observed = TRUE, rxx_restricted = TRUE, ryy_restricted = TRUE)

## Correction for direct range restriction in X
correct_r(correction = "uvdrr_x", rxyi = .3, ux = .8, rxx = .8, ryy = .8,
ux_observed = TRUE, uy_observed = TRUE, rxx_restricted = TRUE, ryy_restricted = TRUE)

## Correction for indirect range restriction in X
correct_r(correction = "uvirr_x", rxyi = .3, ux = .8, rxx = .8, ryy = .8,
ux_observed = TRUE, uy_observed = TRUE, rxx_restricted = TRUE, ryy_restricted = TRUE)

## Correction for direct range restriction in X and Y
correct_r(correction = "bvdrr", rxyi = .3, ux = .8, uy = .8, rxx = .8, ryy = .8,
ux_observed = TRUE, uy_observed = TRUE, rxx_restricted = TRUE, ryy_restricted = TRUE)

## Correction for indirect range restriction in X and Y
correct_r(correction = "bvirr", rxyi = .3, ux = .8, uy = .8, rxx = .8, ryy = .8,
ux_observed = TRUE, uy_observed = TRUE, rxx_restricted = TRUE, ryy_restricted = TRUE)
```
**correct_r_bias**  
*Correct correlations for small-sample bias*

**Description**

Corrects Pearson correlations ($r$) for small-sample bias

**Usage**

```r
correct_r_bias(r, n)
```

**Arguments**

- `r`  
  Vector of correlations.
- `n`  
  Vector of sample sizes.

**Details**

\[
r_c = \frac{r_{obs}}{\left(\frac{2n-2}{2n-1}\right)}
\]

**Value**

Vector of correlations corrected for small-sample bias.

**References**


**Examples**

```r
correct_r_bias(r = .3, n = 30)
correct_r_bias(r = .3, n = 300)
correct_r_bias(r = .3, n = 3000)
```
**Correct correlations for scale coarseness**

**Description**
Corrects correlations for scale coarseness.

**Usage**

```r
correct_r_coarseness(
  r,
  kx = NULL,
  ky = NULL,
  n = NULL,
  dist_x = "norm",
  dist_y = "norm",
  bin_value_x = c("median", "mean", "index"),
  bin_value_y = c("median", "mean", "index"),
  width_x = 3,
  width_y = 3,
  lbound_x = NULL,
  ubound_x = NULL,
  lbound_y = NULL,
  ubound_y = NULL,
  index_values_x = NULL,
  index_values_y = NULL
)
```

**Arguments**

- **r** Observed correlation.
- **kx, ky** Number of scale points used to measure the x and y variables. Set to NULL to treat as continuously measured.
- **n** Optional sample size.
- **dist_x, dist_y** Assumed latent distribution of the x and y variables.
- **bin_value_x, bin_value_y** Are the scale points used to measure the of the x and y variables assumed to represent bin medians, means, or index values?
- **width_x, width_y** For symmetrically distributed variables, how many standard deviations above/below the latent mean should be be used for the latent variable range to make the correction? (Note: Setting width > 3 produces erratic results.) The latent variable range can alternatively be set using lbound and ubound.
- **lbound_x, lbound_y** What lower bound of the range for the latent x and y variables should be used to make the correction? (Note: For normally distributed variables, setting lbound < -3 produces erratic results.)
ubound_x, ubound_y
What upper bound of the range for the latent x and y variables should be used to make the correction? (Note: For normally distributed variables, setting ubound > 3 produces erratic results.)

index_values_x, index_values_y
Optional. If bin_value = "index", the bin index values. If unspecified, values 1:k are used.

Value
Vector of correlations corrected for scale coarseness (if n is supplied, corrected error variance and adjusted sample size is also reported).

References


Examples

```r
correct_r_coarseness(r = .35, kx = 5, ky = 4, n = 100)
correct_r_coarseness(r = .35, kx = 5, n = 100)
correct_r_coarseness(r = .35, kx = 5, ky = 4, n = 100, dist_x="unif", dist_y="norm")
```

```r
correct_r_dich(r, px = NA, py = NA, n = NULL, ...)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>Vector of correlations attenuated by artificial dichomization.</td>
</tr>
<tr>
<td>px</td>
<td>Vector of proportions of the distribution on either side of the split applied to X (set as NA if X is continuous).</td>
</tr>
</tbody>
</table>
**correct_r_split**

- **py** Vector of proportions of the distribution on either side of the split applied to Y (set as NA if Y is continuous).
- **n** Optional vector of sample sizes.
- **...** Additional arguments.

**Details**

\[ r_c = \frac{r_{\text{obs}}}{\frac{\phi(p_X)}{p_X(1-p_X)} - \frac{\phi(p_Y)}{p_Y(1-p_Y)}} \]

**Value**

Vector of correlations corrected for artificial dichomization (if n is supplied, corrected error variance and adjusted sample size is also reported).

**References**


**Examples**

```r
correct_r_dich(r = 0.32, px = .5, py = .5, n = 100)
```

**Description**

This correction is mathematically equivalent to correcting the correlation for direct range restriction in the split variable.

**Usage**

```r
correct_r_split(r, pi, pa = 0.5, n = NULL)
```

**Arguments**

- **r** Vector of correlations affected by an uneven or unrepresentative split of a dichotomous variable.
- **pi** Vector of proportions of incumbent/sample cases in one of the categories of the dichotomous variable.
- **pa** Vector of proportions of applicant/population cases in one of the categories of the dichotomous variable.
- **n** Optional vector of sample sizes.
$r_c = \frac{r_{obs}}{u \sqrt{\left( \frac{1}{u^2} - 1 \right) r_{obs}^2 + 1}}$

where $u = \sqrt{\frac{p_i(1-p_i)}{p_a(1-p_a)}}$, the ratio of the dichotomous variance in the sample ($p_i$ is the incumbent/sample proportion in one of the two groups) to the dichotomous variance in the population ($p_a$ is the applicant/population proportion in one of the two groups). This correction is identical to the correction for univariate direct range restriction, applied to a dichotomous variable.

**Value**

Vector of correlations corrected for unrepresentative splits (if \( n \) is supplied, corrected error variance and adjusted sample size is also reported).

**References**


**Examples**

```r
correct_r_split(r = 0.3, pi = .9, pa = .5, n = 100)
```

**Description**

This function generates artifact-distribution objects containing either interactive or Taylor series artifact distributions. Use this to create objects that can be supplied to the `ma_r_ad` and `ma_r_2_ad` functions to apply psychometric corrections to barebones meta-analysis objects via artifact distribution methods.

Allows consolidation of observed and estimated artifact information by cross-correcting artifact distributions and forming weighted artifact summaries.

For \( u \) ratios, error variances can be computed for independent samples (i.e., settings in which the unrestricted standard deviation comes from an external study) or dependent samples (i.e., settings in which the range-restricted standard deviation comes from a sample that represents a subset of the applicant sample that provided the unrestricted standard deviation). The former circumstance is presumed to be more common, so error variances are computed for independent samples by default.
Usage

```r
create_ad(
  ad_type = c("tsa", "int"),
  rxxi = NULL,
  n_rxxi = NULL,
  wt_rxxi = n_rxxi,
  rxxi_type = rep("alpha", length(rxxi)),
  k_items_rxxi = rep(NA, length(rxxi)),
  rxxa = NULL,
  n_rxxa = NULL,
  wt_rxxa = n_rxxa,
  rxxa_type = rep("alpha", length(rxxa)),
  k_items_rxxa = rep(NA, length(rxxa)),
  ux = NULL,
  ni_ux = NULL,
  na_ux = NULL,
  wt_ux = ni_ux,
  dep_sds_ux_obs = rep(FALSE, length(ux)),
  ut = NULL,
  ni_ut = NULL,
  na_ut = NULL,
  wt_ut = ni_ut,
  dep_sds_ut_obs = rep(FALSE, length(ut)),
  mean_qxi = NULL,
  var_qxi = NULL,
  k_qxi = NULL,
  mean_n_qxi = NULL,
  qxi_dist_type = rep("alpha", length(mean_qxi)),
  mean_k_items_qxi = rep(NA, length(mean_qxi)),
  mean_rxxi = NULL,
  var_rxxi = NULL,
  k_rxxi = NULL,
  mean_n_rxxi = NULL,
  rxxi_dist_type = rep("alpha", length(mean_rxxi)),
  mean_k_items_rxxi = rep(NA, length(mean_rxxi)),
  mean_qxa = NULL,
  var_qxa = NULL,
  k_qxa = NULL,
  mean_n_qxa = NULL,
  qxa_dist_type = rep("alpha", length(mean_qxa)),
  mean_k_items_qxa = rep(NA, length(mean_qxa)),
  mean_rxxa = NULL,
  var_rxxa = NULL,
  k_rxxa = NULL,
  mean_n_rxxa = NULL,
  rxxa_dist_type = rep("alpha", length(mean_rxxa)),
  mean_k_items_rxxa = rep(NA, length(mean_rxxa)),
  mean_ux = NULL,
```
var_ux = NULL,
k_ux = NULL,
mean_ni_ux = NULL,
mean_na_ux = rep(NA, length(mean_ux)),
dep_sds_ux_spec = rep(FALSE, length(mean_ux)),
mean_ut = NULL,
var_ut = NULL,
k_ut = NULL,
mean_ni_ut = NULL,
mean_na_ut = rep(NA, length(mean_ut)),
dep_sds_ut_spec = rep(FALSE, length(mean_ut)),
estimate_rxxa = TRUE,
estimate_rxxi = TRUE,
estimate_ux = TRUE,
estimate_ut = TRUE,
var_unbiased = TRUE,
)

Arguments

ad_type Type of artifact distribution to be computed: Either "tsa" for Taylor series approximation or "int" for interactive.
rxxi Vector of incumbent reliability estimates.
n_rxxi Vector of sample sizes associated with the elements of rxxi.
wt_rxxi Vector of weights associated with the elements of rxxi (by default, sample sizes will be used as weights).
rxxi_type, rxxa_type, qxi_dist_type, rxxi_dist_type, qxa_dist_type, rxxa_dist_type String vector identifying the types of reliability estimates supplied (e.g., "alpha", "retest", "interrater_r", "splithalf"). See the documentation for ma_r for a full list of acceptable reliability types.
k_items_rxxi, mean_k_items_qxi, mean_k_items_rxxi, k_items_rxxa, mean_k_items_qxa, mean_k_items_rxxa Numeric vector of the number of items in each scale (or mean number of items, for pre-specified distributions).
rxxa Vector of applicant reliability estimates.
n_rxxa Vector of sample sizes associated with the elements of rxxa.
wt_rxxa Vector of weights associated with the elements of rxxa (by default, sample sizes will be used as weights).
ux Vector of observed-score u ratios.
ni_ux Vector of incumbent sample sizes associated with the elements of ux.
na_ux Vector of applicant sample sizes that can be used in estimating the sampling error of supplied ux values. NULL by default. Only used when ni_ux is not NULL. If supplied, must be either a scalar or the same length as ni_ux.
wt_ux Vector of weights associated with the elements of ux (by default, sample sizes will be used as weights).
create_ad

- **dep_sds_ux_obs**: Logical scalar or vector determining whether supplied ux values were computed using dependent samples (TRUE) or independent samples (FALSE).
- **ut**: Vector of true-score u ratios.
- **ni_ut**: Vector of incumbent sample sizes associated with the elements of ut.
- **na_ut**: Vector of applicant sample sizes that can be used in estimating the sampling error of supplied ut values. NULL by default. Only used when ni_ut is not NULL. If supplied, must be either a scalar or the same length as ni_ut.
- **wt_ut**: Vector of weights associated with the elements of ut (by default, sample sizes will be used as weights).
- **dep_sds_ut_obs**: Logical scalar or vector determining whether supplied ut values were computed using dependent samples (TRUE) or independent samples (FALSE).
- **mean_qxi**: Vector that can be used to supply the means of externally computed distributions of incumbent square-root reliabilities.
- **var_qxi**: Vector that can be used to supply the variances of externally computed distributions of incumbent square-root reliabilities.
- **k_qxi**: Vector that can be used to supply the number of studies included in externally computed distributions of incumbent square-root reliabilities.
- **mean_n_qxi**: Vector that can be used to supply the mean sample sizes of externally computed distributions of incumbent square-root reliabilities.
- **mean_rxxi**: Vector that can be used to supply the means of externally computed distributions of incumbent reliabilities.
- **var_rxxi**: Vector that can be used to supply the variances of externally computed distributions of incumbent reliabilities.
- **k_rxxi**: Vector that can be used to supply the number of studies included in externally computed distributions of incumbent reliabilities.
- **mean_n_rxxi**: Vector that can be used to supply the mean sample sizes of externally computed distributions of incumbent reliabilities.
- **mean_qxa**: Vector that can be used to supply the means of externally computed distributions of applicant square-root reliabilities.
- **var_qxa**: Vector that can be used to supply the variances of externally computed distributions of applicant square-root reliabilities.
- **k_qxa**: Vector that can be used to supply the number of studies included in externally computed distributions of applicant square-root reliabilities.
- **mean_n_qxa**: Vector that can be used to supply the mean sample sizes of externally computed distributions of applicant square-root reliabilities.
- **mean_rxxa**: Vector that can be used to supply the means of externally computed distributions of applicant reliabilities.
- **var_rxxa**: Vector that can be used to supply the variances of externally computed distributions of applicant reliabilities.
- **k_rxxa**: Vector that can be used to supply the number of studies included in externally computed distributions of applicant reliabilities.
- **mean_n_rxxa**: Vector that can be used to supply the mean sample sizes of externally computed distributions of applicant reliabilities.
Vector that can be used to supply the means of externally computed distributions of observed-score $u$ ratios.

Vector that can be used to supply the variances of externally computed distributions of observed-score $u$ ratios.

Vector that can be used to supply the number of studies included in externally computed distributions of observed-score $u$ ratios.

Vector that can be used to supply the mean incumbent sample sizes of externally computed distributions of observed-score $u$ ratios.

Vector or scalar that can be used to supply the mean applicant sample size(s) of externally computed distributions of observed-score $u$ ratios.

Logical scalar or vector determining whether externally computed $ux$ distributions were computed using dependent samples (TRUE) or independent samples (FALSE).

Vector that can be used to supply the means of externally computed distributions of true-score $u$ ratios.

Vector that can be used to supply the variances of externally computed distributions of true-score $u$ ratios.

Vector that can be used to supply the number of studies included in externally computed distributions of true-score $u$ ratios.

Vector that can be used to supply the mean sample sizes for of externally computed distributions of true-score $u$ ratios.

Vector or scalar that can be used to supply the mean applicant sample size(s) of externally computed distributions of true-score $u$ ratios.

Logical scalar or vector determining whether externally computed $ut$ distributions were computed using dependent samples (TRUE) or independent samples (FALSE).

Logical argument to estimate $rxxa$ values from other artifacts (TRUE) or to only used supplied $rxxa$ values (FALSE). TRUE by default.

Logical argument to estimate $rxxi$ values from other artifacts (TRUE) or to only used supplied $rxxi$ values (FALSE). TRUE by default.

Logical argument to estimate $ux$ values from other artifacts (TRUE) or to only used supplied $ux$ values (FALSE). TRUE by default.

Logical argument to estimate $ut$ values from other artifacts (TRUE) or to only used supplied $ut$ values (FALSE). TRUE by default.

Logical scalar determining whether variance should be unbiased (TRUE) or maximum-likelihood (FALSE).

Further arguments.

Artifact distribution object (matrix of artifact-distribution means and variances) for use artifact-distribution meta-analyses.
Examples

## Example computed using observed values only:
```r
create_ad(ad_type = "tsa", rxxa = c(.9, .8), n_rxxa = c(50, 150),
          rxxi = c(.8, .7), n_rxxi = c(50, 150),
          ux = c(.9, .8), ni_ux = c(50, 150))
create_ad(ad_type = "int", rxxa = c(.9, .8), n_rxxa = c(50, 150),
          rxxi = c(.8, .7), n_rxxi = c(50, 150),
          ux = c(.9, .8), ni_ux = c(50, 150))
```

## Example computed using all possible input arguments (arbitrary values):
```r
rxxa <- rxxi <- ux <- ut <- c(.7, .8)
mean_qxa <- mean_qxi <- mean_ux <- mean_ut <- c(.7, .8)
var_qxa <- var_qxi <- var_ux <- var_ut <- var_rxxa <- var_rxxi <- c(.1, .05)
mean_rxxa <- mean_rxxi <- c(100, 100)
mean_ux <- mean_ut <- c(100, 100)
mean_ni_ux <- mean_ni_ut <- c(100, 100)
dep_sds_ux_obs <- dep_sds_ux_spec <- FALSE
dep_sds_ut_obs <- dep_sds_ut_spec <- FALSE
```

```r
create_ad(rxxa = rxxa, n_rxxa = n_rxxa, wt_rxxa = wt_rxxa,
          mean_qxa = mean_qxa, var_qxa = var_qxa,
          k_qxa = k_qxa, mean_n_qxa = mean_n_qxa,
          mean_rxxa = mean_rxxa, var_rxxa = var_rxxa,
          k_rxxa = k_rxxa, mean_n_rxxa = mean_n_rxxa,
          rxxi = rxxi, n_rxxi = n_rxxi, wt_rxxi = wt_rxxi,
          mean_qxi = mean_qxi, var_qxi = var_qxi,
          k_qxi = k_qxi, mean_n_qxi = mean_n_qxi,
          mean_rxxi = mean_rxxi, var_rxxi = var_rxxi,
          k_rxxi = k_rxxi, mean_n_rxxi = mean_n_rxxi,
          ux = ux, ni_ux = ni_ux, na_ux = na_ux, wt_ux = wt_ux,
          dep_sds_ux_obs = dep_sds_ux_obs,
          mean_ux = mean_ux, var_ux = var_ux, k_ux = k_ux, mean_ni_ux = mean_ni_ux,
          mean_na_ux = mean_na_ux, dep_sds_ux_spec = dep_sds_ux_spec,
          ut = ut, ni_ut = ni_ut, na_ut = na_ut, wt_ut = wt_ut,
          dep_sds_ut_obs = dep_sds_ut_obs)
```
create_ad_group

Generate an artifact distribution object for a dichotomous grouping variable.

Description

This function generates artifact-distribution objects containing either interactive or Taylor series artifact distributions for dichotomous group-membership variables. Use this to create objects that can be supplied to the ma_r_ad and ma_d_ad functions to apply psychometric corrections to barebones meta-analysis objects via artifact distribution methods.

Allows consolidation of observed and estimated artifact information by cross-correcting artifact distributions and forming weighted artifact summaries.

Usage

create_ad_group(
  ad_type = c("tsa", "int"),
  rGg = NULL,
  n_rGg = NULL,
  wt_rGg = n_rGg,
  pi = NULL,
  pa = NULL,
  n_pi = NULL,
  n_pa = NULL,
  wt_p = n_pi,
  mean_rGg = NULL,
  var_rGg = NULL,
  k_rGg = NULL,
  mean_n_rGg = NULL,
  var_unbiased = TRUE,
  ...
)

Arguments

ad_type  Type of artifact distribution to be computed: Either "tsa" for Taylor series approximation or "int" for interactive.

rGg      Vector of incumbent reliability estimates.

n_rGg    Vector of sample sizes associated with the elements of rGg.
create_ad_group

wt_rGg Vector of weights associated with the elements of rGg (by default, sample sizes will be used as weights if provided).

pi Vector of incumbent/sample proportions of members in one of the two groups being compared (one or both of pi/pa can be vectors - if both are vectors, they must be of equal length).

pa Vector of applicant/population proportions of members in one of the two groups being compared (one or both of pi/pa can be vectors - if both are vectors, they must be of equal length).

n_pi Vector of sample sizes associated with the elements in pi.

n_pa Vector of sample sizes associated with the elements in pa.

wt_p Vector of weights associated with the collective element pairs in pi and pa.

mean_rGg Vector that can be used to supply the means of externally computed distributions of correlations between observed and latent group membership.

var_rGg Vector that can be used to supply the variances of externally computed distributions of correlations between observed and latent group membership.

k_rGg Vector that can be used to supply the number of studies included in externally computed distributions of correlations between observed and latent group membership.

mean_n_rGg Vector that can be used to supply the mean sample sizes of externally computed distributions of correlations between observed and latent group membership.

var_unbiased Logical scalar determining whether variance should be unbiased (TRUE) or maximum-likelihood (FALSE).

... Further arguments.

Value

Artifact distribution object (matrix of artifact-distribution means and variances) for use in artifact-distribution meta-analyses.

Examples

## Example artifact distribution for a dichotomous grouping variable:
create_ad_group(rGg = c(.8, .9, .95), n_rGg = c(100, 200, 250),
              mean_rGg = .9, var_rGg = .05,
              k_rGg = 5, mean_n_rGg = 100,
              pi = c(.6, .55, .3), pa = .5, n_pi = c(100, 200, 250), n_pa = c(300, 300, 300),
              var_unbiased = TRUE)

create_ad_group(ad_type = "int", rGg = c(.8, .9, .95), n_rGg = c(100, 200, 250),
              mean_rGg = .9, var_rGg = .05,
              k_rGg = 5, mean_n_rGg = 100,
              pi = c(.6, .55, .3), pa = .5, n_pi = c(100, 200, 250), n_pa = c(300, 300, 300),
              var_unbiased = TRUE)
create_ad_tibble  Create a tibble of artifact distributions by construct

Description
Create a tibble of artifact distributions by construct

Usage

create_ad_tibble(
  ad_type = c("tsa", "int"),
  n = NULL,
  sample_id = NULL,
  construct_x = NULL,
  facet_x = NULL,
  measure_x = NULL,
  construct_y = NULL,
  facet_y = NULL,
  measure_y = NULL,
  rxx = NULL,
  rxx_restricted = TRUE,
  rxx_type = "alpha",
  k_items_x = NA,
  ryy = NULL,
  ryy_restricted = TRUE,
  ryy_type = "alpha",
  k_items_y = NA,
  ux = NULL,
  ux_observed = TRUE,
  uy = NULL,
  uy_observed = TRUE,
  estimate_rxxa = TRUE,
  estimate_rxxi = TRUE,
  estimate_ux = TRUE,
  estimate_ut = TRUE,
  moderators = NULL,
  cat_moderators = TRUE,
  moderator_type = c("simple", "hierarchical", "none"),
  construct_order = NULL,
  supplemental_ads = NULL,
  data = NULL,
  control = control_psychmeta(),
  ...
)

create_ad_list(
  ad_type = c("tsa", "int"),
create_ad_tibble

n = NULL,
sample_id = NULL,
construct_x = NULL,
facet_x = NULL,
measure_x = NULL,
construct_y = NULL,
facet_y = NULL,
measure_y = NULL,
rxx = NULL,
rxx_restricted = TRUE,
rxx_type = "alpha",
k_items_x = NA,
ryy = NULL,
ryy_restricted = TRUE,
ryy_type = "alpha",
k_items_y = NA,
ux = NULL,
ux_observed = TRUE,
uy = NULL,
uy_observed = TRUE,
estimate_rxxa = TRUE,
estimate_rxxi = TRUE,
estimate_ux = TRUE,
estimate_ut = TRUE,
moderators = NULL,
cat_moderators = TRUE,
moderator_type = c("simple", "hierarchical", "none"),
construct_order = NULL,
supplemental_ads = NULL,
data = NULL,
control = control_psychmeta(),
...
)

Arguments

ad_type        Type of artifact distributions to be computed: Either "tsa" for Taylor series approximation or "int" for interactive.
n             Vector or column name of sample sizes.
sample_id      Optional vector of identification labels for samples/studies in the meta-analysis.
construct_x, construct_y Vector of construct names for constructs initially designated as "X" or "Y".
facet_x, facet_y Vector of facet names for constructs initially designated as "X" or "Y". Facet names "global", "overall", and "total" are reserved to indicate observations that represent effect sizes that have already been composited or that represent construct-level measurements rather than facet-level measurements. To avoid double-compositing, any observation with one of these reserved names will only be
create_ad_tibble

eligible for auto-compositing with other such observations and will not be combined with narrow facets.

measure_x, measure_y
Vector of names for measures associated with constructs initially designated as "X" or "Y".

rxx
Vector or column name of reliability estimates for X.

rxx_restricted
Logical vector or column name determining whether each element of rxx is an incumbent reliability (TRUE) or an applicant reliability (FALSE).

rxx_type, ryy_type
String vector identifying the types of reliability estimates supplied. See documentation of ma_r for a full list of acceptable values.

k_items_x, k_items_y
Numeric vector identifying the number of items in each scale.

ryy
Vector or column name of reliability estimates for Y.

ryy_restricted
Logical vector or column name determining whether each element of ryy is an incumbent reliability (TRUE) or an applicant reliability (FALSE).

ux
Vector or column name of u ratios for X.

ux_observed
Logical vector or column name determining whether each element of ux is an observed-score u ratio (TRUE) or a true-score u ratio (FALSE).

uy
Vector or column name of u ratios for Y.

uy_observed
Logical vector or column name determining whether each element of uy is an observed-score u ratio (TRUE) or a true-score u ratio (FALSE).

estimate_rxxa
Logical argument to estimate rxxa values from other artifacts (TRUE) or to only used supplied rxxa values (FALSE). TRUE by default.

estimate_rxxi
Logical argument to estimate rxxi values from other artifacts (TRUE) or to only used supplied rxxi values (FALSE). TRUE by default.

estimate_ux
Logical argument to estimate ux values from other artifacts (TRUE) or to only used supplied ux values (FALSE). TRUE by default.

estimate_ut
Logical argument to estimate ut values from other artifacts (TRUE) or to only used supplied ut values (FALSE). TRUE by default.

moderators
Matrix or column names of moderator variables to be used in the meta-analysis (can be a vector in the case of one moderator).

cat_moderators
Logical scalar or vector identifying whether variables in the moderators argument are categorical variables (TRUE) or continuous variables (FALSE).

moderator_type
Type of moderator analysis: "none" means that no moderators are to be used, "simple" means that moderators are to be examined one at a time, and "hierarchical" means that all possible combinations and subsets of moderators are to be examined.

construct_order
Vector indicating the order in which variables should be arranged, with variables listed earlier in the vector being preferred for designation as X.
supplemental_ads

Named list (named according to the constructs included in the meta-analysis) of supplemental artifact distribution information from studies not included in the meta-analysis. This is a list of lists, where the elements of a list associated with a construct are named like the arguments of the create_ad() function.

data

Data frame containing columns whose names may be provided as arguments to vector arguments.

control

Output from the control_psychmeta() function or a list of arguments controlled by the control_psychmeta() function. Ellipsis arguments will be screened for internal inclusion in control.

Value

A tibble of artifact distributions

Examples

```r
## Examples to create Taylor series artifact distributions:
# Overall artifact distributions (not pairwise, not moderated)
create_ad_tibble(ad_type = "tsa",
                 n = n, rxx = rxxi, ryy = ryyi,
                 construct_x = x_name, construct_y = y_name,
                 sample_id = sample_id, moderators = moderator,
                 data = data_r_meas_multi,
                 control = control_psychmeta(pairwise_ads = FALSE,
                                              moderated_ads = FALSE))

# Overall artifact distributions by moderator combination
create_ad_tibble(ad_type = "tsa",
                 n = n, rxx = rxxi, ryy = ryyi,
                 construct_x = x_name, construct_y = y_name,
                 sample_id = sample_id, moderators = moderator,
                 data = data_r_meas_multi,
                 control = control_psychmeta(pairwise_ads = FALSE,
                                              moderated_ads = TRUE))

# Pairwise artifact distributions (not moderated)
create_ad_tibble(ad_type = "tsa",
                 n = n, rxx = rxxi, ryy = ryyi,
                 construct_x = x_name, construct_y = y_name,
                 sample_id = sample_id, moderators = moderator,
                 data = data_r_meas_multi,
                 control = control_psychmeta(pairwise_ads = TRUE,
                                              moderated_ads = FALSE))

# Pairwise artifact distributions by moderator combination
create_ad_tibble(ad_type = "tsa",
                 n = n, rxx = rxxi, ryy = ryyi,
                 construct_x = x_name, construct_y = y_name,
                 sample_id = sample_id, moderators = moderator,
                 data = data_r_meas_multi,
                 control = control_psychmeta(pairwise_ads = TRUE,
                                              moderated_ads = FALSE))
```
credibility

Description

Function to construct a credibility interval around a mean effect size.

Usage

credibility(mean, sd, k = NULL, cred_level = 0.8, cred_method = c("t", "norm"))

Arguments

mean
Mean effect size.

sd
Residual/true standard deviation of effect sizes, after accounting for variance from artifacts.

k
Number of studies in the meta-analysis.

cred_level
Credibility level that defines the width of the credibility interval (default = 0.8).

cred_method
Distribution to be used to compute the width of credibility intervals. Available options are "t" for t distribution or "norm" for normal distribution.

Details

$$CR = mean_{es} \pm \text{quantile} \times SD_{es}$$

Value

A matrix of credibility intervals of the specified width.

Examples

credibility(mean = .3, sd = .15, cred_level = .8, cred_method = "norm")
credibility(mean = .3, sd = .15, cred_level = .8, k = 10)
credibility(mean = c(.3, .5), sd = c(.15, .2), cred_level = .8, k = 10)
**data_d_bb_multi**

*Hypothetical d value dataset simulated with sampling error only*

**Description**

Data set for use in example meta-analyses of multiple variables.

**Usage**

```r
data(data_d_bb_multi)
```

**Format**

data.frame

**Examples**

```r
data(data_d_bb_multi)
```

**data_d_meas_multi**

*Hypothetical d value dataset simulated to satisfy the assumptions of the correction for measurement error only in multiple constructs*

**Description**

Data set for use in example meta-analyses correcting for measurement error in multiple variables.

**Usage**

```r
data(data_d_meas_multi)
```

**Format**

data.frame

**Examples**

```r
data(data_d_meas_multi)
```
data_r_bvirr

**Hypothetical dataset simulated to satisfy the assumptions of the bivariate correction for direct range restriction**

**Description**

Data set for use in example meta-analyses of bivariate direct range restriction. Note that the BVDRR correction is only an approximation of the appropriate range-restriction correction and tends to have a noticeable positive bias when applied in meta-analyses.

**Usage**

```r
data(data_r_bvirr)
```

**Format**

data.frame

**Examples**

```r
data(data_r_bvirr)
```

---

data_r_bvirr

**Hypothetical dataset simulated to satisfy the assumptions of the bivariate correction for indirect range restriction**

**Description**

Data set for use in example meta-analyses of bivariate indirect range restriction.

**Usage**

```r
data(data_r_bvirr)
```

**Format**

data.frame

**Examples**

```r
data(data_r_bvirr)
```
**data_r_gonzalezmule_2014**  
*Meta-analysis of OCB correlations with other constructs*

**Description**
Data set to demonstrate corrections for univariate range restriction and measurement error using individual corrections or artifact distributions. NOTE: This is an updated version of the data set reported in the Gonzalez-Mulé, Mount, an Oh (2014) article that was obtained from the first author.

**Usage**
```r
data(data_r_gonzalezmule_2014)
```

**Format**
data.frame

**References**

**Examples**
```r
data(data_r_gonzalezmule_2014)
```

**data_r_mcdaniel_1994**  
*Artifact-distribution meta-analysis of the validity of interviews*

**Description**
Data set to demonstrate corrections for univariate range restriction and criterion measurement error using artifact distributions.

**Usage**
```r
data(data_r_mcdaniel_1994)
```

**Format**
data.frame
References

Examples

```r
data(data_r_mcdaniel_1994)
```

---

**data_r_mcleod_2007** Bare-bones meta-analysis of parenting and childhood depression

Description
Data set to demonstrate bare-bones meta-analysis.

Usage

```r
data(data_r_mcleod_2007)
```

Format
data.frame

References

Examples

```r
data(data_r_mcleod_2007)
```

---

**data_r_meas** Hypothetical dataset simulated to satisfy the assumptions of the correction for measurement error only

Description
Data set for use in example meta-analyses correcting for measurement error in two variables.

Usage

```r
data(data_r_meas)
```
**data_r_meas_multi**

**Format**
data.frame

**Examples**
data(data_r_meas)

data_r_meas_multi  
*Hypothetical correlation dataset simulated to satisfy the assumptions of the correction for measurement error only in multiple constructs*

**Description**
Data set for use in example meta-analyses correcting for measurement error in multiple variables.

**Usage**
data(data_r_meas_multi)

**Format**
data.frame

**Examples**
data(data_r_meas_multi)

data_r_oh_2009  
*Second order meta-analysis of operational validities of big five personality measures across East Asian countries*

**Description**
Example of a second-order meta-analysis of correlations corrected using artifact-distribution methods.

**Usage**
data(data_r_oh_2009)

**Format**
data.frame
References


Examples

data(data_r_oh_2009)

data(data_r_roth_2015)

---

data_r_roth_2015  Artifact-distribution meta-analysis of the correlation between school grades and cognitive ability

Description

Data set to demonstrate corrections for univariate range restriction and cognitive ability measurement error.

Usage

data(data_r_roth_2015)

Format

data.frame

References


Examples

data(data_r_roth_2015)
data_r_uvdrr

Hypothetical dataset simulated to satisfy the assumptions of the univariate correction for direct range restriction

Description
Data set for use in example meta-analyses correcting for univariate direct range restriction.

Usage
data(data_r_uvdrr)

Format
data.frame

Examples
data(data_r_uvdrr)

data_r_uvirr

Hypothetical dataset simulated to satisfy the assumptions of the univariate correction for indirect range restriction

Description
Data set for use in example meta-analyses correcting for univariate indirect range restriction.

Usage
data(data_r_uvirr)

Format
data.frame

Examples
data(data_r_uvirr)
Estimation of applicant and incumbent reliabilities and of true- and observed-score $u$ ratios

Description

Functions to estimate the values of artifacts from other artifacts. These functions allow for reliability estimates to be corrected/attenuated for range restriction and allow $u$ ratios to be converted between observed-score and true-score metrics. Some functions also allow for the extrapolation of an artifact from other available information.

Available functions include:

- **estimate_rxxa**
  Estimate the applicant reliability of variable $X$ from $X$’s incumbent reliability value and $X$’s observed-score or true-score $u$ ratio.

- **estimate_rxxa_u**
  Estimate the applicant reliability of variable $X$ from $X$’s observed-score and true-score $u$ ratios.

- **estimate_rxxi**
  Estimate the incumbent reliability of variable $X$ from $X$’s applicant reliability value and $X$’s observed-score or true-score $u$ ratio.

- **estimate_rxxi_u**
  Estimate the incumbent reliability of variable $X$ from $X$’s observed-score and true-score $u$ ratios.

- **estimate_ux**
  Estimate the true-score $u$ ratio for variable $X$ from $X$’s reliability coefficient and $X$’s observed-score $u$ ratio.

- **estimate_uy**
  Estimate the observed-score $u$ ratio for variable $X$ from $X$’s reliability coefficient and $X$’s true-score $u$ ratio.

- **estimate_ryya**
  Estimate the applicant reliability of variable $Y$ from $Y$’s incumbent reliability value, $Y$’s correlation with $X$, and $X$’s $u$ ratio.

- **estimate_ryyi**
  Estimate the incumbent reliability of variable $Y$ from $Y$’s applicant reliability value, $Y$’s correlation with $X$, and $X$’s $u$ ratio.

- **estimate_uy**
  Estimate the observed-score $u$ ratio for variable $Y$ from $Y$’s applicant and incumbent reliability coefficients.

- **estimate_up**
  Estimate the true-score $u$ ratio for variable $Y$ from $Y$’s applicant and incumbent reliability coefficients.
Usage

```r
estimate_rxxa(
  rxxi,
  ux,
  ux_observed = TRUE,
  indirect_rr = TRUE,
  rxxi_type = "alpha"
)

estimate_rxxi(
  rxxa,
  ux,
  ux_observed = TRUE,
  indirect_rr = TRUE,
  rxxa_type = "alpha"
)

estimate_ut(ux, rxx, rxx_restricted = TRUE)

estimate_ux(ut, rxx, rxx_restricted = TRUE)

estimate_ryya(ryyi, rxyi, ux)

estimate_ryyi(ryya, rxyi, ux)

estimate_uy(ryyi, ryya, indirect_rr = TRUE, ryy_type = "alpha")

estimate_up(ryyi, ryya)

estimate_rxxa_u(ux, ut)

estimate_rxxi_u(ux, ut)
```

Arguments

- **rxxi**: Vector of incumbent reliability estimates for X.
- **ux**: Vector of observed-score u ratios for X (if used in the context of estimating a reliability value, a true-score u ratio may be supplied by setting `ux_observed` to `FALSE`).
- **ux_observed**: Logical vector determining whether each element of `ux` is an observed-score u ratio (TRUE) or a true-score u ratio (FALSE).
- **indirect_rr**: Logical vector determining whether each reliability value is associated with indirect range restriction (TRUE) or direct range restriction (FALSE).
- **rxxi_type**, **rxxa_type**, **ryy_type**: String vector identifying the types of reliability estimates supplied (e.g., "alpha", "retest", "interrater_r", "splithalf"). See the documentation for `ma_r` for a full list of acceptable reliability types.
estimate_artifacts

\(r_{xxa}\) Vector of applicant reliability estimates for X.

\(r_{xx}\) Vector of reliability estimates for X (used in the context of estimating \(u_X\) and \(u_T\) - specify that reliability is an incumbent value by setting \(r_{xx\_restricted}\) to FALSE).

\(r_{xx\_restricted}\) Logical vector determining whether each element of \(r_{xx}\) is an incumbent reliability (TRUE) or an applicant reliability (FALSE).

\(u_T\) Vector of true-score \(u\) ratios for X.

\(r_{yyi}\) Vector of incumbent reliability estimates for Y.

\(r_{xi}\) Vector of observed-score incumbent correlations between X and Y.

\(r_{yya}\) Vector of applicant reliability estimates for Y.

Details

#### Formulas to estimate \(r_{xxa}\) ####

Formulas for indirect range restriction:

\[
\rho_{XXa} = 1 - u_X^2 (1 - \rho_{XXi})
\]

\[
\rho_{XXa} = \frac{\rho_{XXi}}{\rho_{XXi} + u_T^2 - \rho_{XXi} u_T^2}
\]

Formula for direct range restriction:

\[
\rho_{XXa} = \frac{\rho_{XXi}}{u_X^2 \left[1 + \rho_{XXi} \left(\frac{1}{u_X^2} - 1\right)\right]}
\]

#### Formulas to estimate \(r_{xxi}\) ####

Formulas for indirect range restriction:

\[
\rho_{XXi} = 1 - \frac{1 - \rho_{XXa}}{u_X^2}
\]

\[
\rho_{XXi} = 1 - \frac{1 - \rho_{XXa}}{\rho_{XXa} \left[u_T^2 - (1 - \frac{1}{\rho_{XXa}})\right]}
\]

Formula for direct range restriction:

\[
\rho_{XXi} = \frac{\rho_{XXa} u_X^2}{1 + \rho_{XXa} (u_X^2 - 1)}
\]

#### Formulas to estimate \(u_T\) ####

\[
u_T = \sqrt{\frac{\rho_{XXa} u_X^2}{1 + \rho_{XXi} u_X^2 - u_X^2}}
\]

\[
u_T = \sqrt{\frac{u_T^2 - (1 - \rho_{XXa})}{\rho_{XXa}}}
\]
### Formulas to estimate $u_X$ ###

\[
u_X = \sqrt{\frac{u_T^2}{\rho_{XX_a} \left( 1 + \frac{u_T^2}{\rho_{XX_a}} - \frac{1}{\rho_{XX_a}} \right)}}
\]

\[
u_X = \sqrt{\rho_{XX_a} \left[ u_T^2 - \left( 1 - \frac{1}{\rho_{XX_a}} \right) \right]}
\]

### Formula to estimate $\rho_{YY_a}$ ###

\[
\rho_{YY_a} = 1 - \frac{1 - \rho_{YY_i}}{1 - \rho_{XY_i} \left( 1 - \frac{1}{u_X^2} \right)}
\]

### Formula to estimate $\rho_{YY_i}$ ###

\[
\rho_{YY_i} = 1 - (1 - \rho_{YY_a}) \left[ 1 - \rho_{XY_i} \left( 1 - \frac{1}{u_X^2} \right) \right]
\]

### Formula to estimate $u_Y$ ###

\[
u_Y = \sqrt{\frac{1 - \rho_{YY_a}}{1 - \rho_{YY_i}}}
\]

### Formula to estimate $u_P$ ###

\[
u_P = \sqrt{\frac{1 - \rho_{YY_a}}{\rho_{YY_a} - (1 - \rho_{YY_a})}}
\]

**Value**

A vector of estimated artifact values.

**References**


**Examples**

estimate_rxxa(rxxi = .8, ux = .8, ux_observed = TRUE)
estimate_rxxi(rxxa = .8, ux = .8, ux_observed = TRUE)
estimate_ut(ux = .8, rxx = .8, rxx_restricted = TRUE)
estimate_ux(ut = .8, rxx = .8, rxx_restricted = TRUE)
estimate_ryya(ryyi = .8, rxyi = .3, ux = .8)
estimate_ryyi(ryya = .8, rxyi = .3, ux = .8)
estimate_uy(ryyi = c(.5, .7), ryya = c(.7, .8))
estimate_up(ryyi = c(.5, .7), ryya = c(.7, .8))
estimate_rxxa_u(ux = c(.7, .8), ut = c(.65, .75))
estimate_rxxi_u(ux = c(.7, .8), ut = c(.65, .75))

---

**estimate_length_sb**  
Inverse Spearman-Brown formula to estimate the amount by which a measure would have to be lengthened or shorted to achieve a desired level of reliability

**Description**

This function implements the inverse of the Spearman-Brown prophecy formula and answers the question: "How much would I have to increase (or decrease) the length of this measure to obtain a desired reliability level given the current reliability of the measure?" The result of the function is the multiplier by which the length of the original measure should be adjusted. The formula implemented here assumes that all items added to (or subtracted from) the measure will be parallel forms of the original items.

**Usage**

estimate_length_sb(rel_initial, rel_desired)

**Arguments**

rel_initial  
Initial reliability of a measure.

rel_desired  
Desired reliability of a lengthened or shortened measure.

**Details**

This is computed as:

\[ k^* = \frac{\rho_{XX}^* (\rho_{XX} - 1)}{(\rho_{XX}^* - 1) \rho_{XX}} \]

where \( \rho_{XX} \) is the initial reliability, \( \rho_{XX}^* \) is the predicted reliability of a measure with a different length, and \( k^* \) is the number of times the measure would have to be lengthened to obtain a reliability equal to \( \rho_{XX}^* \).
The estimated number of times by which the number of items in the initial measure would have to be multiplied to achieve the desired reliability.

References


Examples

```r
# Estimated k to achieve a reliability of .8 from a measure with an initial reliability of .7
estimate_length_sb(rel_initial = .7, rel_desired = .8)
```

```r
# Estimated k to achieve a reliability of .8 from a measure with an initial reliability of .9
estimate_length_sb(rel_initial = .9, rel_desired = .8)
```

---

**estimate_prod**  
*Estimation of statistics computed from products of random, normal variables*

**Description**

This family of functions computes univariate descriptive statistics for the products of two variables denoted as "x" and "y" (e.g., mean(x * y) or var(x * y)) and the covariance between the products of "x" and "y" and of "u" and "v" (e.g., cov(x * y, u * v) or cor(x * y, u * v)). These functions presume all variables are random normal variables.

Available functions include:

- `estimate_mean_prod`
  Estimate the mean of the product of two variables: x * y.

- `estimate_var_prod`
  Estimate the variance of the product of two variables: x * y.

- `estimate_cov_prods`
  Estimate the covariance between the products of two pairs of variables: x * y and u * v.

- `estimate_cor_prods`
  Estimate the correlation between the products of two pairs of variables: x * y and u * v.

**Usage**

```r
estimate_mean_prod(mu_x, mu_y, cov_xy)
```

```r
estimate_var_prod(mu_x, mu_y, var_x, var_y, cov_xy)
```

```r
estimate_cov_prods(mu_x, mu_y, mu_u, mu_v, cov_xu, cov_xv, cov_yu, cov_yv)
```
estimate_cor_prods(
    mu_x,
    mu_y,
    mu_u,
    mu_v,
    var_x,
    var_y,
    var_u,
    var_v,
    cov_xu,
    cov_xv,
    cov_yu,
    cov_yv,
    cov_xy,
    cov_uv
)

Arguments

mu_x             Expected value of variable x.
mu_y             Expected value of variable y.
cov_xy           Covariance between x and y.
var_x            Variance of variable x.
var_y            Variance of variable y.
mu_u             Expected value of variable u.
mu_v             Expected value of variable v.
cov_xu           Covariance between x and u.
cov_xv           Covariance between x and v.
cov_yu           Covariance between y and u.
cov_yv           Covariance between y and v.
var_u            Variance of variable u.
var_v            Variance of variable v.
cov_uv           Covariance between u and v.

Value

An estimated statistic computed from the products of random, normal variables.

References


**estimate_q_dist**

*Estimate descriptive statistics of square-root reliabilities*

**Description**

Estimate descriptive statistics of square-root reliabilities from descriptive statistics of reliabilities via Taylor series approximation.

**Usage**

```r
estimate_q_dist(mean_rel, var_rel)
```

**Arguments**

- `mean_rel`: Mean reliability value.
- `var_rel`: Variance of reliability values.

**Details**

\[
\text{var}_{qX} = \frac{\text{var}_{\rho_{XX}}}{4q_X^2}
\]

**Value**

The estimated mean and variance of a distribution of square-root reliability values.

**Examples**

```r
estimate_q_dist(mean_rel = .8, var_rel = .15)
```

---

**estimate_rel_dist**

*Estimate descriptive statistics of reliabilities*

**Description**

Estimate descriptive statistics of reliabilities from descriptive statistics of square-root reliabilities via Taylor series approximation.

**Usage**

```r
estimate_rel_dist(mean_q, var_q)
```

**Arguments**

- `mean_q`: Mean square-root reliability value.
- `var_q`: Variance of square-root reliability values.
**Details**

\[ \text{var}_{\rho_{XX}} = 4q^2_{X} \text{var}_{\rho_{XX}} \]

**Value**

The estimated mean and variance of a distribution of reliability values.

**Examples**

```r
estimate_rel_dist(mean_q = .9, var_q = .05)
```

**estimate_rel_sb**  
*Spearman-Brown prophecy formula to estimate the reliability of a lengthened measure*

**Description**

This function implements the Spearman-Brown prophecy formula for estimating the reliability of a lengthened (or shortened) measure. The formula implemented here assumes that all items added to (or subtracted from) the measure will be parallel forms of the original items.

**Usage**

```r
estimate_rel_sb(rel_initial, k)
```

**Arguments**

- `rel_initial`  
  Initial reliability of a measure.

- `k`  
  The number of times by which the measure should be lengthened (if \( k > 1 \)) or shortened (if \( k < 1 \)), assuming that all new items are parallel forms of initial items.

**Details**

This is computed as:

\[
\rho_{XX}^* = \frac{k \rho_{XX}}{1 + (k - 1) \rho_{XX}}
\]

where \( \rho_{XX} \) is the initial reliability, \( k \) is the multiplier by which the measure is to be lengthened (or shortened), and \( \rho_{XX}^* \) is the predicted reliability of a measure with a different length.

**Value**

The estimated reliability of the lengthened (or shortened) measure.
References


Examples

```r
## Double the length of a measure with an initial reliability of .7
estimate_rel_sb(rel_initial = .7, k = 2)

## Halve the length of a measure with an initial reliability of .9
estimate_rel_sb(rel_initial = .9, k = .5)
```

### estimate_u

*Estimate u ratios from available artifact information*

#### Description

Uses information about standard deviations, reliability estimates, and selection ratios to estimate u ratios. Selection ratios are only used to estimate u when no other information is available, but estimates of u computed from SDs and reliabilities will be averaged to reduce error.

#### Usage

```r
estimate_u(
  measure_id = NULL,
  sdi = NULL,
  sda = NULL,
  rxxi = NULL,
  rxxa = NULL,
  item_ki = NULL,
  item_ka = NULL,
  n = NULL,
  meani = NULL,
  sr = NULL,
  rxya_est = NULL,
  data = NULL
)
```

#### Arguments

- `measure_id`: Vector of measure identifiers.
- `sdi`: Scalar or vector containing restricted standard deviation(s).
- `sda`: Scalar or vector containing unrestricted standard deviation(s).
- `rxxi`: Scalar or vector containing restricted reliability coefficient(s).
- `rxxa`: Scalar or vector containing unrestricted reliability coefficient(s).
item_ki  Scalar or vector containing the number of items used in measures within samples.

item_ka  Scalar or vector indicating the number of items toward which reliability estimates should be adjusted using the Spearman-Brown formula.

n        Vector of sample sizes.

meani    Vector of sample means.

sr       Vector of selection ratios (used only when no other useable u-ratio inputs are available).

rxya_est Vector of estimated unrestricted correlations between the selection mechanism and the variable of interest (used only when sr is used).

data     Optional data frame containing any or all information for use in other arguments.

Value

A vector of estimated u ratios.

Examples

sdi <- c(1.4, 1.2, 1.3, 1.4)
sda <- 2
rxxi <- c(.6, .7, .75, .8)
rxxa <- c(.9, .95, .8, .9)
item_ki <- c(12, 12, 12, NA)
item_ka <- NULL
n <- c(200, 200, 200, 200)
meani <- c(2, 1, 2, 3)
sr <- c(.5, .6, .7, .4)
rxya_est <- .5

## Estimate u from standard deviations only:
estimate_u(sdi = sdi, sda = sda)

## Estimate u from incumbent standard deviations and the mixture standard deviation:
estimate_u(sdi = sdi, sda = "mixture", meani = meani, n = n)

## Estimate u from reliability information:
estimate_u(rxxi = rxxi, rxxa = rxxa)

## Estimate u from both standard deviations and reliabilities:
estimate_u(sdi = sdi, sda = sda, rxxi = rxxi, rxxa = rxxa, item_ki = item_ki, item_ka = item_ka, n = n, meani = meani, sr = sr, rxya_est = rxya_est)

estimate_u(sdi = sdi, sda = "average", rxxi = rxxi, rxxa = "average", item_ki = item_ki, item_ka = item_ka, n = n, meani = meani)

## Estimate u from selection ratios as direct range restriction:
estimate_u(sr = sr)
## Estimate u from selection ratios as indirect range restriction:
estimate_u(sr = sr, rxya_est = rxya_est)

---

**estimate_var_artifacts**

*Taylor series approximations for the variances of estimates artifact distributions.*

### Description

Taylor series approximations to estimate the variances of artifacts that have been estimated from other artifacts. These functions are implemented internally in the `create_ad` function and related functions, but are useful as general tools for manipulating artifact distributions.

Available functions include:

- **estimate_var_qxi**  
  Estimate the variance of a qxi distribution from a qxa distribution and a distribution of u ratios.

- **estimate_var_rxxi**  
  Estimate the variance of an rxxi distribution from an rxxa distribution and a distribution of u ratios.

- **estimate_var_qxa**  
  Estimate the variance of a qxa distribution from a qxi distribution and a distribution of u ratios.

- **estimate_var_rxxa**  
  Estimate the variance of an rxxa distribution from an rxxi distribution and a distribution of u ratios.

- **estimate_var_ut**  
  Estimate the variance of a true-score u ratio distribution from an observed-score u ratio distribution and a reliability distribution.

- **estimate_var_ux**  
  Estimate the variance of an observed-score u ratio distribution from a true-score u ratio distribution and a reliability distribution.

- **estimate_var_qyi**  
  Estimate the variance of a qyi distribution from the following distributions: qya, rxyi, and ux.

- **estimate_var_ryyi**  
  Estimate the variance of an ryyi distribution from the following distributions: ryya, rxyi, and ux.

- **estimate_var_qya**  
  Estimate the variance of a qya distribution from the following distributions: qyi, rxyi, and ux.

- **estimate_var_ryya**  
  Estimate the variance of an ryya distribution from the following distributions: ryyi, rxyi, and ux.
Usage

```r
estimate_var_qxi(
  qxa,
  var_qxa = 0,
  ux,
  var_ux = 0,
  cor_qxa_ux = 0,
  ux_observed = TRUE,
  indirect_rr = TRUE,
  qxa_type = "alpha"
)
```

```r
estimate_var_qxa(
  qxi,
  var_qxi = 0,
  ux,
  var_ux = 0,
  cor_qxi_ux = 0,
  ux_observed = TRUE,
  indirect_rr = TRUE,
  qxi_type = "alpha"
)
```

```r
estimate_var_rxxi(
  rxxa,
  var_rxxa = 0,
  ux,
  var_ux = 0,
  cor_rxxa_ux = 0,
  ux_observed = TRUE,
  indirect_rr = TRUE,
  rxxa_type = "alpha"
)
```

```r
estimate_var_rxxa(
  rxxi,
  var_rxxi = 0,
  ux,
  var_ux = 0,
  cor_rxxi_ux = 0,
  ux_observed = TRUE,
  indirect_rr = TRUE,
  rxxi_type = "alpha"
)
```

```r
estimate_var_ut(
  rxx,
  var_rxx = 0,
  ```
estimate_var_artifacts

ux,
var_ux = 0,
cor_rxx_ux = 0,
rxx_restricted = TRUE,
rxx_as_qx = FALSE
)

estimate_var_ux(
  rxx,
  var_rxx = 0,
  ut,
  var_ut = 0,
  cor_rxx_ut = 0,
  rxx_restricted = TRUE,
  rxx_as_qx = FALSE
)

estimate_var_ryya(
  ryyi,
  var_ryyi = 0,
  rxyi,
  var_rxyi = 0,
  ux,
  var_ux = 0,
  cor_ryyi_rxyi = 0,
  cor_ryyi_ux = 0,
  cor_rxyi_ux = 0
)

estimate_var_qya(
  qyi,
  var_qyi = 0,
  rxyi,
  var_rxyi = 0,
  ux,
  var_ux = 0,
  cor_qyi_rxyi = 0,
  cor_qyi_ux = 0,
  cor_rxyi_ux = 0
)

estimate_var_qyi(
  qya,
  var_qya = 0,
  rxyi,
  var_rxyi = 0,
  ux,
  var_ux = 0,
cor_qya_rxyi = 0,
cor_qya_ux = 0,
cor_rxyi_ux = 0
)

estimate_var_ryyi(
  ryya,
  var_ryya = 0,
  rxyi,
  var_rxyi = 0,
  ux,
  var_ux = 0,
  cor_ryya_rxyi = 0,
  cor_ryya_ux = 0,
  cor_rxyi_ux = 0
)

Arguments

qxa          Square-root of applicant reliability estimate.
var_qxa      Variance of square-root of applicant reliability estimate.
ux           Observed-score u ratio.
var_ux       Variance of observed-score u ratio.
cor_qxa_ux   Correlation between qxa and ux.
ux_observed  Logical vector determining whether u ratios are observed-score u ratios (TRUE) or true-score u ratios (FALSE).
indirect_rr  Logical vector determining whether reliability values are associated with indirect range restriction (TRUE) or direct range restriction (FALSE).
qxi          Square-root of incumbent reliability estimate.
var_qxi      Variance of square-root of incumbent reliability estimate.
cor_qxi_ux   Correlation between qxi and ux.
rxxa         Incumbent reliability value.
var_rxxa     Variance of incumbent reliability values.
cor_rxxa_ux  Correlation between rxxa and ux.
rxxi         Incumbent reliability value.
var_rxxi     Variance of incumbent reliability values.
cor_rxxi_ux  Correlation between rxxi and ux.
rxxi_type, rxxa_type, qx1_type, qx_a_type
  String vector identifying the types of reliability estimates supplied (e.g., "alpha", "retest", "intrater_r", "splithalf"). See the documentation for ma_r for a full list of acceptable reliability types.
rxx          Generic argument for a reliability estimate (whether this is a reliability or the square root of a reliability is clarified by the rxx_as_qx argument).
var_rxx       Generic argument for the variance of reliability estimates (whether this pertains to reliabilities or the square roots of reliabilities is clarified by the rxx_as_qx argument).
cor_rxx_ux    Correlation between rxx and ux.
rxx_restricted Logical vector determining whether reliability estimates were incumbent reliabilities (TRUE) or applicant reliabilities (FALSE).
rxx_as_qx     Logical vector determining whether the reliability estimates were reliabilities (TRUE) or square-roots of reliabilities (FALSE).
ut          True-score u ratio.
var_ut        Variance of true-score u ratio.
cor_rxx_ut    Correlation between rxx and ut.
ryyi      Incumbent reliability value.
var_ryyi      Variance of incumbent reliability values.
rxyi      Incumbent correlation between X and Y.
var_rxyi      Variance of incumbent correlations.
cor_ryyi_rxyi Correlation between ryyi and rxyi.
cor_ryyi_ux   Correlation between ryyi and ux.
cor_rxyi_ux   Correlation between rxyi and ux.
qyi          Square-root of incumbent reliability estimate.
var_qyi       Variance of square-root of incumbent reliability estimate.
cor_qyi_rxyi  Correlation between qyi and rxyi.
cor_qyi_ux    Correlation between qyi and ux.
qya          Square-root of applicant reliability estimate.
var_qya       Variance of square-root of applicant reliability estimate.
cor_qya_rxyi  Correlation between qya and rxyi.
cor_qya_ux    Correlation between qya and ux.
ryya         Applicant reliability value.
var_ryya      Variance of applicant reliability values.
cor_ryya_rxyi Correlation between ryya and rxyi.
cor_ryya_ux   Correlation between ryya and ux.

Details

#### Partial derivatives to estimate the variance of qxa using ux ####

Indirect range restriction:

\[
b_{u_X} = \frac{(q_{X_i}^2 - 1)u_X}{\sqrt{(q_{X_i}^2 - 1)u_X^2 + 1}}
\]

\[
b_{q_{X_i}} = \frac{q_{X_i}^2u_X^2}{\sqrt{(q_{X_i}^2 - 1)u_X^2 + 1}}
\]
Direct range restriction:

\[
b_{u_X} = \frac{q_X^2(q_X^2 - 1)u_X}{\sqrt{-\frac{q_X^2}{q_X^2(u_X^2-1)} - u_X^2(q_X^2(u_X^2-1) - u_X^2)^2}} \\
b_{q_X} = \frac{q_Xu_X^2}{\sqrt{-\frac{q_X^2}{q_X^2(u_X^2-1)} - u_X^2(q_X^2(u_X^2-1) - u_X^2)^2}}
\]

#### Partial derivatives to estimate the variance of \(r_{xxa}\) using \(u_X\) ####

Indirect range restriction:

\[
b_{u_X} = 2(\rho_{XX} - 1)u_X \\
\rho_{XX} : b_{\rho_{XX}} = u_X^2
\]

Direct range restriction:

\[
b_{u_X} = \frac{2(\rho_{XX} - 1)\rho_{XX}u_X}{(-\rho_{XX}u_X^2 + \rho_{XX} + u_X^2)^2} \\
b_{\rho_{XX}} = \frac{u_X^2}{(-\rho_{XX}u_X^2 + \rho_{XX} + u_X^2)^2}
\]

#### Partial derivatives to estimate the variance of \(r_{xxa}\) using \(u_T\) ####

\[
b_{u_T} = \frac{2(\rho_{XX} - 1)\rho_{XX}u_T}{(-\rho_{XX}u_T^2 + \rho_{XX} + u_T^2)^2} \\
b_{\rho_{XX}} = \frac{u_T^2}{(-\rho_{XX}u_T^2 + \rho_{XX} + u_T^2)^2}
\]

#### Partial derivatives to estimate the variance of \(q_{xa}\) using \(u_T\) ####

\[
b_{u_T} = \frac{q_X^2(q_X^2 - 1)u_T}{\sqrt{-\frac{q_X^2}{q_X^2(u_T^2-1)} - u_T^2(q_X^2(u_T^2-1) - u_T^2)^2}} \\
b_{q_X} = \frac{q_Xu_T^2}{\sqrt{-\frac{q_X^2}{q_X^2(u_T^2-1)} - u_T^2(q_X^2(u_T^2-1) - u_T^2)^2}}
\]

#### Partial derivatives to estimate the variance of \(q_{xi}\) using \(u_X\) ####

Indirect range restriction:

\[
b_{u_X} = \frac{1 - qxu^2}{u_X^3} \sqrt{\frac{q_X^2 + u_X^2 - 1}{u_X^2}} \\
b_{q_X} = \frac{q_X}{u_X^2} \sqrt{\frac{q_Xu^2 - 1}{u_X^2} + 1}
\]
Direct range restriction:

\[ b_{u_X} = -\frac{q_X^2 (q_X^2 - 1) u_X}{\sqrt{q_X^2 u_X^2} (q_X^2 (u_X^2 - 1) + 1)^2} \]
\[ b_{q_X a} = \frac{q_X u_X^2}{\sqrt{q_X^2 u_X^2} (q_X^2 (u_X^2 - 1) + 1)^2} \]

#### Partial derivatives to estimate the variance of rxxi using ux ####

Indirect range restriction:

\[ b_{u_X} = \frac{2 - 2 \rho_{XX a}}{u_X^2} \]
\[ b_{\rho_{XX a}} = \frac{1}{u_X^2} \]

Direct range restriction:

\[ b_{u_X} = -\frac{2 (\rho_{XX a} - 1) \rho_{XX a} u_X}{(\rho_{XX a} (u_X^2 - 1) + 1)^2} \]
\[ b_{\rho_{XX a}} = \frac{u_X^2}{(\rho_{XX a} (u_X^2 - 1) + 1)^2} \]

#### Partial derivatives to estimate the variance of rxxi using ut ####

\[ u_T : b_{u_T} = -\frac{2 (\rho_{XX a} - 1) \rho_{XX a} u_T}{(\rho_{XX a} (u_T^2 - 1) + 1)^2} \]
\[ b_{\rho_{XX a}} = \frac{u_T^2}{(\rho_{XX a} (u_T^2 - 1) + 1)^2} \]

#### Partial derivatives to estimate the variance of qxi using ut ####

\[ b_{u_T} = -\frac{(q_X a - 1) q_X^2 (q_X a + 1) u_T}{\sqrt{q_X^2 u_T^2} (q_X^2 (u_T^2 - q_X^2 a + 1)^2 + 1)^2} \]
\[ b_{q_X a} = \frac{q_X a u_T^2}{\sqrt{q_X^2 u_T^2} (q_X^2 (u_T^2 - q_X^2 a + 1)^2 + 1)^2} \]

#### Partial derivatives to estimate the variance of ut using qxi ####

\[ b_{u_X} = \frac{q_X^2 u_X}{\sqrt{(q_X^2 - 1) u_X^2 + 1} ((q_X^2 - 1) u_X^2 + 1)^2} \]
\[ b_{uX} = -\frac{u_X^2 (u_X^2 - 1)}{\sqrt{\frac{\rho_{XX} u_X^2}{\rho_{XX} - 1} (u_X^2 - 1) u_X^2 + 1}^2} \]

#### Partial derivatives to estimate the variance of \( uT \) using \( r_{XX} \) ####

\[ b_{uT} = \left( u_T^2 - q_X (u_T^2 - 1) \right)^{1.5} \]

\[ b_{qX} = \frac{q_X (u_T^2 - 1)}{u_T^2 (u_T^2 - q_X (u_T^2 - 1)^2} \]

#### Partial derivatives to estimate the variance of \( uX \) using \( q_{Xi} \) ####

\[ b_{uX} = \frac{u_X}{\rho_{XX} u_X} \sqrt{\frac{\rho_{XX} u_X^2}{\rho_{XX} - 1} (u_X^2 - 1) u_X^2 + 1}^2 \]

\[ b_{q_{Xi}} = \frac{1 - u_X^2}{\sqrt{\frac{\rho_{XX} u_X^2}{\rho_{XX} - 1} (u_X^2 - 1) u_X^2 + 1}^2} \]
$b_{\rho_{XX_i}} = \frac{(u_T^2 - 1) \left( \frac{u_i^2}{\rho_{XX_i} + u_T^2} \right)^{1.5}}{2u_T^2}$

##### Partial derivatives to estimate the variance of $u_X$ using $q_Xa$ #####

\[b_{u_X} = \frac{q_X^2(u_T - 1)}{\sqrt{q_X^2(u_T^2 - 1) + 1}}\]

\[b_{q_Xa} = \frac{q_X(u_T - 1)}{\sqrt{q_X^2(u_T^2 - 1) + 1}}\]

##### Partial derivatives to estimate the variance of $u_X$ using $r_{XXa}$ #####

\[b_{u_X} = \frac{\rho_{XXa}u_T}{\sqrt{\rho_{XXa}(u_T^2 - 1) + 1}}\]

\[b_{\rho_{XXa}} = \frac{u_i^2 - 1}{2\sqrt{\rho_{XXa}(u_T^2 - 1) + 1}}\]

##### Partial derivatives to estimate the variance of $r_{YYa}$ #####

\[b_{\rho_{YYi}} = \frac{1}{\rho_{YYi} \left( \frac{1}{u_X^2} - 1 \right) + 1}\]

\[b_{u_X} = \frac{2(\rho_{YYi} - 1)\rho_{XYi}u_X}{(u_X^2 - \rho_{XYi}(u_X^2 - 1))^2}\]

\[b_{\rho_{XYi}} = \frac{2(\rho_{YYi} - 1)\rho_{XYi}u_X^3(u_X^2 - 1)}{(u_X^2 - \rho_{XYi}(u_X^2 - 1))^2}\]

##### Partial derivatives to estimate the variance of $qYa$ #####

\[b_{q_Yi} = \frac{q_Y_i}{\left[ 1 - \rho_{XYi} \left( \frac{1}{u_X^2} \right) \right] \sqrt{1 - \frac{1-q_Y_i}{1-\rho_{XYi} \left( 1-\frac{1}{u_X^2} \right)}}}\]

\[b_{u_X} = -\frac{(1-q_Y_i)\rho_{XYi}}{u_X^3 \left[ 1 - \rho_{XYi} \left( \frac{1}{u_X^2} \right) \right] \sqrt{1 - \frac{1-q_Y_i}{1-\rho_{XYi} \left( 1-\frac{1}{u_X^2} \right)}}}\]

\[b_{\rho_{XYi}} = -\frac{(1-q_Y_i)\rho_{XYi} \left( \frac{1}{u_X^2} \right)}{\left[ 1 - \rho_{XYi} \left( \frac{1}{u_X^2} \right) \right] \sqrt{1 - \frac{1-q_Y_i}{1-\rho_{XYi} \left( 1-\frac{1}{u_X^2} \right)}}}\]

##### Partial derivatives to estimate the variance of $r_{YYi}$ #####


\[ \rho_{YY_a} : b_{\rho_{YY_a}} = \rho_{XY_i}^2 \left( \frac{1}{u_X^2} - 1 \right) + 1 \]

\[ b_{u_X} = -\frac{2(\rho_{YY_a} - 1)\rho_{XY_i}}{u_X^3} \]

\[ b_{\rho_{XY_i}} = -\frac{2(\rho_{YY_a} - 1)\rho_{XY_i} (u_X^2 - 1)}{u_X^3} \]

### Partial derivatives to estimate the variance of \( q_i \) ###

\[ b_{q_{Y_a}} = \frac{q_{Y_a} \left[ 1 - \rho_{XY_i}^2 \left( 1 - \frac{1}{u_X^2} \right) \right]}{\sqrt{1 - (1 - q_{Y_a}) \left[ 1 - \rho_{XY_i}^2 \left( 1 - \frac{1}{u_X^2} \right) \right]}} \]

\[ b_{u_X} = \frac{(1 - q_{Y_a}^2)\rho_{XY_i} \left( 1 - \frac{1}{u_X^2} \right)}{\sqrt{1 - (1 - q_{Y_a}) \left[ 1 - \rho_{XY_i}^2 \left( 1 - \frac{1}{u_X^2} \right) \right]}} \]

\[ b_{\rho_{XY_i}} = \frac{(1 - q_{Y_a}^2)\rho_{XY_i}^2}{u_X^3 \sqrt{1 - (1 - q_{Y_a}) \left[ 1 - \rho_{XY_i}^2 \left( 1 - \frac{1}{u_X^2} \right) \right]}} \]

**Examples**

```r
estimate_var_qxi(qxa = c(.8, .85, .9, .95), var_qxa = c(.02, .03, .04, .05),
ux = .8, var_ux = 0,
ux_observed = c(TRUE, TRUE, FALSE, FALSE),
indirect_rr = c(TRUE, FALSE, TRUE, FALSE))
estimate_var_qxa(qxi = c(.8, .85, .9, .95), var_qxi = c(.02, .03, .04, .05),
ux = .8, var_ux = 0,
ux_observed = c(TRUE, TRUE, FALSE, FALSE),
indirect_rr = c(TRUE, FALSE, TRUE, FALSE))
estimate_var_rxxi(rxxa = c(.8, .85, .9, .95),
var_rxxa = c(.02, .03, .04, .05), ux = .8, var_ux = 0,
ux_observed = c(TRUE, TRUE, FALSE, FALSE),
in indirect_rr = c(TRUE, FALSE, TRUE, FALSE))
estimate_var_rxxa(rxxi = c(.8, .85, .9, .95), var_rxxi = c(.02, .03, .04, .05),
ux = .8, var_ux = 0,
ux_observed = c(TRUE, TRUE, FALSE, FALSE),
in indirect_rr = c(TRUE, FALSE, TRUE, FALSE))
estimate_var_ut(rxx = c(.8, .85, .9, .95), var_rxx = 0,
ux = c(.8, .8, .9, .9), var_ux = c(.02, .03, .04, .05),
rxx_restricted = c(TRUE, TRUE, FALSE, FALSE),
rxx_as_qx = c(TRUE, FALSE, TRUE, FALSE))
estimate_var_ux(rxx = c(.8, .85, .9, .95), var_rxx = 0,
ux = c(.8, .8, .9, .9), var_ux = c(.02, .03, .04, .05),
rxx_restricted = c(TRUE, TRUE, FALSE, FALSE),
rxx_as_qx = c(TRUE, FALSE, TRUE, FALSE))
```
**estimate_var_rho_int**  
Non-linear estimate of variance of $\rho$ corrected for psychometric artifacts using numeric integration

### Description

Functions to estimate the variance of $\rho$ corrected for psychometric artifacts. These functions integrate over the residual distribution of correlations from an interactive artifact-distribution meta-analysis to non-linearly estimate the variance of $\rho$.

Available functions include:

- **estimate_var_rho_int_meas**
  Variance of $\rho$ corrected for measurement error only

- **estimate_var_rho_int_uvdrr**
  Variance of $\rho$ corrected for univariate direct range restriction (i.e., Case II) and measurement error

- **estimate_var_rho_int_bvdrr**
  Variance of $\rho$ corrected for bivariate direct range restriction and measurement error

- **estimate_var_rho_int_uvirr**
  Variance of $\rho$ corrected for univariate indirect range restriction (i.e., Case IV) and measurement error

- **estimate_var_rho_int_bvirr**
  Variance of $\rho$ corrected for bivariate indirect range restriction (i.e., Case V) and measurement error

- **estimate_var_rho_int_rb**
  Variance of $\rho$ corrected using Raju and Burke’s correction for direct range restriction and measurement error

### Usage

```r
estimate_var_rho_int_meas(mean_qx, mean_qy, var_res)

estimate_var_rho_int_uvdrr(
  mean_rxyi, 
  mean_rtpa, 
  mean_qxa, 
  mean_qyi, 
  mean_ux, 
  var_res
)
```
estimate_var_rho_int(uvirr(
  mean_rxyi,
  mean_rtpa,
  mean_qxi,
  mean_qyi,
  mean_ut,
  var_res
))

estimate_var_rho_int(bvirr(mean_qxa, mean_qya, mean_ux, mean_uy, var_res))

estimate_var_rho_int(bvdr(t(mean_rxyi,
  mean_rtpa,
  mean_qxa,
  mean_qya,
  mean_ux,
  mean_uy,
  var_res)
)

estimate_var_rho_int(rb(mean_rxyi,
  mean_rtpa,
  mean_qx,
  mean_qy,
  mean_ux,
  var_res)
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean_qx</td>
<td>Mean square root of reliability for X.</td>
</tr>
<tr>
<td>mean_qy</td>
<td>Mean square root of reliability for Y.</td>
</tr>
<tr>
<td>var_res</td>
<td>Residual variance from an interative artifact distribution (i.e., variance of observed correlations minus predicted error variance and predicted artifact variance).</td>
</tr>
<tr>
<td>mean_rxyi</td>
<td>Mean observed correlation.</td>
</tr>
<tr>
<td>mean_rtpa</td>
<td>Mean corrected correlation.</td>
</tr>
<tr>
<td>mean_qxa</td>
<td>Mean square root of unrestricted reliability for X.</td>
</tr>
<tr>
<td>mean_qyi</td>
<td>Mean square root of restricted reliability for Y.</td>
</tr>
<tr>
<td>mean_ux</td>
<td>Mean observed-score u ratio for X.</td>
</tr>
<tr>
<td>mean_qxi</td>
<td>Mean square root of restricted reliability for X.</td>
</tr>
<tr>
<td>mean_ut</td>
<td>Mean true-score u ratio for X.</td>
</tr>
<tr>
<td>mean_qya</td>
<td>Mean square root of unrestricted reliability for Y.</td>
</tr>
<tr>
<td>mean_uy</td>
<td>Mean observed-score u ratio for Y.</td>
</tr>
</tbody>
</table>
Value

A vector of non-linear estimates of the variance of rho.

Notes

estimate_var_rho_int_meas and estimate_var_rho_int_bvirr do not make use of numeric integration because they are linear functions.

References


Description

Functions to estimate the variance of $\rho$ corrected for psychometric artifacts. These functions use Taylor series approximations (i.e., the delta method) to estimate the variance in observed effect sizes predictable from the variance in artifact distributions based on the partial derivatives.

The available Taylor-series functions include:

- **estimate_var_rho_tsa_meas**
  Variance of $\rho$ corrected for measurement error only

- **estimate_var_rho_tsa_uvdrr**
  Variance of $\rho$ corrected for univariate direct range restriction (i.e., Case II) and measurement error

- **estimate_var_rho_tsa_bvdrr**
  Variance of $\rho$ corrected for bivariate direct range restriction and measurement error

- **estimate_var_rho_tsa_uvirr**
  Variance of $\rho$ corrected for univariate indirect range restriction (i.e., Case IV) and measurement error

- **estimate_var_rho_tsa_bvirr**
  Variance of $\rho$ corrected for bivariate indirect range restriction (i.e., Case V) and measurement error

- **estimate_var_rho_tsa_rb1**
  Variance of $\rho$ corrected using Raju and Burke’s TSA1 correction for direct range restriction and measurement error

- **estimate_var_rho_tsa_rb2**
  Variance of $\rho$ corrected using Raju and Burke’s TSA2 correction for direct range restriction and measurement error. Note that a typographical error in Raju and Burke’s article has been corrected in this function so as to compute appropriate partial derivatives.
Usage

 estimate_var_rho_tsa_meas(
     mean_rtp,
     var_rxy,
     var_e,
     mean_qx = 1,
     var_qx = 0,
     mean_qy = 1,
     var_qy = 0,
     ...
 )

estimate_var_rho_tsa_uvdrr(
     mean_rtpa,
     var_rxyi,
     var_e,
     mean_ux = 1,
     var_ux = 0,
     mean_qxa = 1,
     var_qxa = 0,
     mean_qyi = 1,
     var_qyi = 0,
     ...
 )

estimate_var_rho_tsa_bvdrr(
     mean_rtpa,
     var_rxyi,
     var_e = 0,
     mean_ux = 1,
     var_ux = 0,
     mean_uy = 1,
     var_uy = 0,
     mean_qxa = 1,
     var_qxa = 0,
     mean_qya = 1,
     var_qya = 0,
     ...
 )

estimate_var_rho_tsa_uvirr(
     mean_rtpa,
     var_rxyi,
     var_e,
     mean_ut = 1,
     var_ut = 0,
     mean_qxa = 1,
     var_qxa = 0,
mean_qyi = 1,
var_qyi = 0,
...
)

estimate_var_rho_tsa_bvrirr(
  mean_rtpa,
  var_rxyi,
  var_e = 0,
  mean_ux = 1,
  var_ux = 0,
  mean_uy = 1,
  var_uy = 0,
  mean_qxa = 1,
  var_qxa = 0,
  mean_qya = 1,
  var_qya = 0,
  sign_rxz = 1,
  sign_ryz = 1,
  ...
)

estimate_var_rho_tsa_rb1(
  mean_rtpa,
  var_rxyi,
  var_e,
  mean_ux = 1,
  var_ux = 0,
  mean_rxx = 1,
  var_rxx = 0,
  mean_ryy = 1,
  var_ryy = 0,
  ...
)

estimate_var_rho_tsa_rb2(
  mean_rtpa,
  var_rxyi,
  var_e,
  mean_ux = 1,
  var_ux = 0,
  mean_qx = 1,
  var_qx = 0,
  mean_qy = 1,
  var_qy = 0,
  ...
)
Arguments

- mean_rtp: Mean corrected correlation.
- var_rxy: Variance of observed correlations.
- var_e: Error variance of observed correlations.
- mean_qx: Mean square root of reliability for X.
- var_qx: Variance of square roots of reliability estimates for X.
- mean_qy: Mean square root of reliability for Y.
- var_qy: Variance of square roots of reliability estimates for Y.
- ... Additional arguments.
- mean_rtpa: Mean corrected correlation.
- var_rxyi: Variance of observed correlations.
- mean_ux: Mean observed-score u ratio for X.
- var_ux: Variance of observed-score u ratios for X.
- mean_qxa: Mean square root of unrestricted reliability for X.
- var_qxa: Variance of square roots of unrestricted reliability estimates for X.
- mean_qyi: Mean square root of restricted reliability for Y.
- var_qyi: Variance of square roots of restricted reliability estimates for Y.
- mean_uy: Mean observed-score u ratio for Y.
- var_uy: Variance of observed-score u ratios for Y.
- mean_qya: Mean square root of unrestricted reliability for Y.
- var_qya: Variance of square roots of unrestricted reliability estimates for Y.
- mean_ut: Mean true-score u ratio for X.
- var_ut: Variance of true-score u ratios for X.
- sign_rxz: Sign of the relationship between X and the selection mechanism.
- sign_ryz: Sign of the relationship between Y and the selection mechanism.
- mean_rxx: Mean reliability for X.
- var_rxx: Variance of reliability estimates for X.
- mean_ryy: Mean reliability for Y.
- var_ryy: Variance of reliability estimates for Y.

Details

############ Measurement error only ############

The attenuation formula for measurement error is

\[ \rho_{XY} = \rho_{TP} q_X q_Y \]

where \( \rho_{XY} \) is an observed correlation, \( \rho_{TP} \) is a true-score correlation, and \( q_X \) and \( q_Y \) are the square roots of reliability coefficients for X and Y, respectively.
The Taylor series approximation of the variance of $\rho_{TP}$ can be computed using the following linear equation,

$$ var_{\rho_{TP}} \approx \left[ var_{rxv} - var_e - (b_1^2 var_{qx} + b_2^2 var_{qy}) \right] / b_3^2 $$

where $b_1$, $b_2$, and $b_3$ are first-order partial derivatives of the attenuation formula with respect to $q_X$, $q_Y$, and $\rho_{TP}$, respectively. The first-order partial derivatives of the attenuation formula are:

$$ b_1 = \frac{\partial \rho_{XY}}{\partial q_X} = \rho_{TP} q_Y $$
$$ b_2 = \frac{\partial \rho_{XY}}{\partial q_Y} = \rho_{TP} q_X $$
$$ b_3 = \frac{\partial \rho_{XY}}{\partial \rho_{TP}} = q_X q_Y $$

Univariate direct range restriction (UVDRR; i.e., Case II)

The UVDRR attenuation procedure may be represented as

$$ \rho_{XY} = \frac{\rho_{TP} q_Y q_X u_X}{\sqrt{\rho_{TP}^2 q_X^2 (u_X^2 - 1) + 1}} $$

The attenuation formula can also be represented as:

$$ \rho_{XY} = \rho_{TP} q_Y q_X u_X A $$

where

$$ A = \frac{1}{\sqrt{\rho_{TP}^2 q_X^2 (u_X^2 - 1) + 1}} $$

The Taylor series approximation of the variance of $\rho_{TP}$ can be computed using the following linear equation,

$$ var_{\rho_{TP}} \approx \left[ var_{rxv} - var_e - (b_1^2 var_{qx} + b_2^2 var_{qy} + b_3^2 var_{ux}) \right] / b_4^2 $$

where $b_1$, $b_2$, $b_3$, and $b_4$ are first-order partial derivatives of the attenuation formula with respect to $q_{Xa}$, $q_{Ya}$, $u_X$, and $\rho_{TPa}$, respectively. The first-order partial derivatives of the attenuation formula are:

$$ b_1 = \frac{\partial \rho_{XY}}{\partial q_{Xa}} = \rho_{TPa} q_{Ya} u_X A^3 $$
$$ b_2 = \frac{\partial \rho_{XY}}{\partial q_{Ya}} = \rho_{XY} q_{Ya} $$
$$ b_3 = -\rho_{TPa} q_{Ya} q_{Xa} (\rho_{TPa}^2 q_{Xa}^2 - 1) A^3 $$
\[ b_4 = \frac{\partial \rho_{XY_i}}{\partial \rho_{TP_a}} = q_Y q_{X_a} u_X A^3 \]

Univariate indirect range restriction (UVIRR; i.e., Case IV)
Under univariate indirect range restriction, the attenuation formula yielding \( \rho_{XY_i} \) is:

\[ \rho_{XY_i} = \frac{u_T q_{X_a}}{\sqrt{u_T^2 q_{X_a}^2 + 1 - q_{X_a}^2}} \frac{u_T \rho_{TP_a}}{\sqrt{u_T^2 \rho_{TP_a}^2 + 1 - \rho_{TP_a}^2}} \]

The attenuation formula can also be represented as:

\[ \rho_{XY_i} = q_{X_a} q_Y \rho_{TP_a} u_T^2 A B \]

where

\[ A = \frac{1}{\sqrt{u_T^2 q_{X_a}^2 + 1 - q_{X_a}^2}} \]

and

\[ B = \frac{1}{\sqrt{u_T^2 \rho_{TP_a}^2 + 1 - \rho_{TP_a}^2}} \]

The Taylor series approximation of the variance of \( \rho_{TP_a} \) can be computed using the following linear equation,

\[ \text{var}_{\rho_{TP_a}} \approx \left[ \text{var}_{r_{XY}} - \text{var}_e - (b_1^2 \text{var}_{q_{X_a}} + b_2^2 \text{var}_{q_Y} + b_3^2 \text{var}_{u_T}) \right] / b_4^2 \]

where \( b_1, b_2, b_3, \) and \( b_4 \) are first-order partial derivatives of the attenuation formula with respect to \( q_{X_a}, q_Y, u_T, \) and \( \rho_{TP_a} \), respectively. The first-order partial derivatives of the attenuation formula are:

\[ b_1 = \frac{\partial \rho_{XY_i}}{\partial q_{X_a}} = \frac{\rho_{XY_i}}{q_{X_a}} - \rho_{XY_i} q_{X_a} B^2 \left( u_T^2 - 1 \right) \]

\[ b_2 = \frac{\partial \rho_{XY_i}}{\partial q_Y} = \frac{\rho_{XY_i}}{q_Y} \]

\[ b_3 = \frac{\partial \rho_{XY_i}}{\partial u_T} = \frac{2 \rho_{XY_i}}{u_T} - \rho_{XY_i} u_T q_{X_a}^2 B^2 - \rho_{XY_i} u_T \rho_{TP_a} A^2 \]

\[ b_4 = \frac{\partial \rho_{XY_i}}{\partial \rho_{TP_a}} = \frac{\rho_{XY_i}}{\rho_{TP_a}} - \rho_{XY_i} \rho_{TP_a} A^2 \left( u_T^2 - 1 \right) \]

Bivariate direct range restriction (BVDRR)
Under bivariate direct range restriction, the attenuation formula yielding \( \rho_{XY_i} \) is:
\[
\rho_{XY_i} = \frac{A + \rho_{TP_a}^2 q_X a q_Y a - \frac{1}{q_X a q_Y a}}{2 \rho_{TP_a} u_X u_Y}
\]

where

\[
A = \sqrt{\left( \frac{1}{q_X a q_Y a} - \rho_{TP_a}^2 q_X a q_Y a \right)^2 + 4 \rho_{TP_a}^2 u_X^2 u_Y^2}
\]

The Taylor series approximation of the variance of \(\rho_{TP_a}\) can be computed using the following linear equation,

\[
\text{var}_{\rho_{TP_a}} \approx [\text{var}_{r_{XY_i}} - \text{var}_e - (b_1^2 \text{var}_{q_X a} + b_2^2 \text{var}_{q_Y a} + b_3^2 \text{var}_{u_X} + b_4^2 \text{var}_{u_Y})] / b_5^2
\]

where \(b_1, b_2, b_3, b_4,\) and \(b_5\) are first-order partial derivatives of the attenuation formula with respect to \(q_X a, q_Y a, u_X, u_Y,\) and \(\rho_{TP_a},\) respectively. First, we define terms to simplify the computation of partial derivatives:

\[
B = (\rho_{TP_a}^2 q_X^2 q_Y^2 a + q_X a q_Y a A - 1)
\]

\[
C = 2 \rho_{TP_a} q_X^2 q_Y^2 a u_X u_Y A
\]

The first-order partial derivatives of the attenuation formula are:

\[
b_1 = \frac{\partial \rho_{XY_i}}{\partial q_X a} = \frac{\left(\rho_{TP_a}^2 q_X^2 q_Y a + 1\right) B}{q_X a C}
\]

\[
b_2 = \frac{\partial \rho_{XY_i}}{\partial q_Y a} = \frac{\left(\rho_{TP_a}^2 q_X^2 q_Y^2 + 1\right) B}{q_Y a C}
\]

\[
b_3 = \frac{\partial \rho_{XY_i}}{\partial u_X} = \frac{\left(\rho_{TP_a} q_X a q_Y a - 1\right) \left(\rho_{TP_a} q_X a q_Y a + 1\right) B}{u_X C}
\]

\[
b_4 = \frac{\partial \rho_{XY_i}}{\partial u_Y} = \frac{\left(\rho_{TP_a} q_X a q_Y a - 1\right) \left(\rho_{TP_a} q_X a q_Y a + 1\right) B}{u_Y C}
\]

\[
b_5 = \frac{\partial \rho_{XY_i}}{\partial \rho_{TP_a}} = \frac{\left(\rho_{TP_a}^2 q_X^2 q_Y a + 1\right) B}{\rho_{TP_a} C}
\]

######## Bivariate indirect range restriction (BIVRR; i.e., Case V) ########

Under bivariate indirect range restriction, the attenuation formula yielding \(\rho_{XY_i}\) is:

\[
\rho_{XY_i} = \rho_{TP_a} q_X a q_Y a - \lambda \sqrt{|1 - u_X^2| |1 - u_Y^2|}
\]

The Taylor series approximation of the variance of \(\rho_{TP_a}\) can be computed using the following linear equation,
\[ \text{var}_{\rho TP_a} \approx [\text{var}_{rXY} - var_e - (b_1^2 \text{var}_{qX} + b_2^2 \text{var}_{qY} + b_3^2 \text{var}_{uX} + b_4^2 \text{var}_{uY})] / b_5^2 \]

where \( b_1, b_2, b_3, b_4, \) and \( b_5 \) are first-order partial derivatives of the attenuation formula with respect to \( qX, qY, uX, uY, \) and \( \rho TP_a \), respectively. First, we define terms to simplify the computation of partial derivatives:

\[
\begin{align*}
    b_1 &= \frac{\partial \rho_{XY_i}}{\partial qX} = \frac{\rho_{TP_a} qY_a}{uX uY} \\
    b_2 &= \frac{\partial \rho_{XY_i}}{\partial qY} = \frac{\rho_{TP_a} qX_a}{uX uY} \\
    b_3 &= \frac{\partial \rho_{XY_i}}{\partial uX} = \frac{\lambda (1 - uX^2)}{uY |1 - uX^2|^{1.5}} \frac{\rho_{XY_i}}{uX} \\
    b_4 &= \frac{\partial \rho_{XY_i}}{\partial uY} = \frac{\lambda (1 - uY^2)}{uX |1 - uY^2|^{1.5}} \frac{\rho_{XY_i}}{uY} \\
    b_5 &= \frac{\partial \rho_{XY_i}}{\partial \rho_{TP_a}} = \frac{qX_a qY_a}{uX uY}
\end{align*}
\]

Raju and Burke’s TSA1 procedure

Raju and Burke’s attenuation formula may be represented as

\[
\rho_{XY} = \frac{\rho_{TP_a} uX \sqrt{\rho_{XX} \rho_{YY}}}{\sqrt{\rho_{TP_a}^2 \rho_{XX} \rho_{YY} uX^2 - \rho_{TP_a}^2 \rho_{XX} \rho_{YY} + 1}}
\]

The Taylor series approximation of the variance of \( \rho_{TP_a} \) can be computed using the following linear equation,

\[
\text{var}_{\rho TP_a} \approx [\text{var}_{rXY} - var_e - (B^2 \text{var}_{\rho YY} + C^2 \text{var}_{\rho XX} + D^2 \text{var}_{uX})] / A^2
\]

where \( A, B, C, \) and \( D \) are first-order partial derivatives of the attenuation formula with respect to \( \rho_{TP_a}, \rho_{XX}, \rho_{YY}, \) and \( uX \), respectively. The first-order partial derivatives of the attenuation formula are:

\[
\begin{align*}
    A &= \frac{\partial \rho_{XY_i}}{\partial \rho_{TP_a}} = \frac{\rho_{XY_i}}{\rho_{TP_a}} + \frac{\rho_{XY_i}^3 (1 - uX^2)}{\rho_{TP_a} uX^2} \\
    B &= \frac{\partial \rho_{XY_i}}{\partial \rho_{YY}} = \frac{1}{2} \left( \frac{\rho_{XY_i}}{\rho_{YY}} + \frac{\rho_{XY_i}^3 (1 - uX^2)}{\rho_{YY} uX^2} \right) \\
    C &= \frac{\partial \rho_{XY_i}}{\partial \rho_{XX}} = \frac{1}{2} \left( \frac{\rho_{XY_i}}{\rho_{XX}} + \frac{\rho_{XY_i}^3 (1 - uX^2)}{\rho_{XX} uX^2} \right)
\end{align*}
\]
\[ D = \frac{\partial \rho_{XY}}{\partial u_X} = \frac{\rho_{XY} - \rho_{XY}^3}{u_X} \]

Raju and Burke's TSA2 procedure

Raju and Burke's attenuation formula may be represented as

\[ \rho_{XY} = \frac{\rho_{TP_a}q_Xa q_Ya u_X}{\sqrt{\rho_{TP_a}^2 q_Xa^2 q_Ya^2 u_X^2 - \rho_{TP_a}^2 q_Xa^2 q_Ya^2}} + 1 \]

The Taylor series approximation of the variance of \( \rho_{TP_a} \) can be computed using the following linear equation,

\[ \text{var}_{\rho_{TP_a}} \approx [\text{var}_{\rho_{XY}} - \text{var}_{\epsilon} - (E^2 \text{var}_{q_Y} + G^2 \text{var}_{q_X} + H^2 \text{var}_{u_X})] / E^2 \]

where E, F, G, and H are first-order partial derivatives of the attenuation formula with respect to \( \rho_{TP_a}, q_X, q_Y, \) and \( u_X, \) respectively. The first-order partial derivatives of the attenuation formula (with typographic errors in the original article corrected) are:

\[
\begin{align*}
E &= \frac{\partial \rho_{XY}}{\partial \rho_{TP_a}} = \frac{\rho_{XY}}{\rho_{TP_a}} + \frac{\rho_{XY}^3(1-u_X^2)}{\rho_{TP_a} u_X^2} \\
F &= \frac{\partial \rho_{XY}}{\partial q_Y} = \frac{\rho_{XY}}{q_Y} + \frac{\rho_{XY}^3(1-u_X^2)}{q_Y u_X^2} \\
G &= \frac{\partial \rho_{XY}}{\partial q_X} = \frac{\rho_{XY}}{q_X} + \frac{\rho_{XY}^3(1-u_X^2)}{q_X u_X^2} \\
H &= \frac{\partial \rho_{XY}}{\partial u_X} = \frac{\rho_{XY} - \rho_{XY}^3}{u_X}
\end{align*}
\]

Value

Vector of meta-analytic variances estimated via Taylor series approximation.

Notes

A typographical error in Raju and Burke's article has been corrected in estimate_var_rho_tsa_rb2 so as to compute appropriate partial derivatives.

References


Examples

estimate_var_rho_tsa_meas(mean_rtp = .5, var_rxy = .02, var_e = .01,
mean_qx = .8, var_qx = .005,
mean_qy = .8, var_qy = .005)
estimate_var_rho_tsa_uvdrr(mean_rtpa = .5, var_rxyi = .02, var_e = .01,
mean_ux = .8, var_ux = .005,
mean_qxa = .8, var_qxa = .005,
mean_qyi = .8, var_qyi = .005)
estimate_var_rho_tsa_bvdrr(mean_rtpa = .5, var_rxyi = .02, var_e = .01,
mean_ux = .8, var_ux = .005,
mean_uy = .8, var_uy = .005,
mean_qxa = .8, var_qxa = .005,
mean_qya = .8, var_qya = .005)
estimate_var_rho_tsa_uvrr(mean_rtpa = .5, var_rxyi = .02, var_e = .01,
mean_ut = .8, var_ut = .005,
mean_qxa = .8, var_qxa = .005,
mean_qyi = .8, var_qyi = .005)
estimate_var_rho_tsa_bvrr(mean_rtpa = .5, var_rxyi = .02, var_e = .01,
mean_ux = .8, var_ux = .005,
mean_uy = .8, var_uy = .005,
mean_qxa = .8, var_qxa = .005,
mean_qya = .8, var_qya = .005,
sign_rxz = 1, sign_ryz = 1)
estimate_var_rho_tsa_rb1(mean_rtpa = .5, var_rxyi = .02, var_e = .01,
mean_ux = .8, var_ux = .005,
mean_rxx = .8, var_rxx = .005)
estimate_var_rho_tsa_rb2(mean_rtpa = .5, var_rxyi = .02, var_e = .01,
mean_ux = .8, var_ux = .005,
mean_qx = .8, var_qx = .005,
mean_qy = .8, var_qy = .005)

estimate_var_tsa

Taylor Series Approximation of effect-size variances corrected for psychometric artifacts

Description

Functions to estimate the variances corrected for psychometric artifacts. These functions use Taylor series approximations (i.e., the delta method) to estimate the corrected variance of an effect-size distribution.

The available Taylor-series functions include:

- **estimate_var_tsa_meas**
  Variance of ρ corrected for measurement error only

- **estimate_var_tsa_uvdrr**
  Variance of ρ corrected for univariate direct range restriction (i.e., Case II) and measurement error
estimate_var_tsa

- `estimate_var_tsa_bvdrr`
  Variance of $\rho$ corrected for bivariate direct range restriction and measurement error

- `estimate_var_tsa_uvirr`
  Variance of $\rho$ corrected for univariate indirect range restriction (i.e., Case IV) and measurement error

- `estimate_var_tsa_bvirr`
  Variance of $\rho$ corrected for bivariate indirect range restriction (i.e., Case V) and measurement error

- `estimate_var_tsa_rb1`
  Variance of $\rho$ corrected using Raju and Burke’s TSA1 correction for direct range restriction and measurement error

- `estimate_var_tsa_rb2`
  Variance of $\rho$ corrected using Raju and Burke’s TSA2 correction for direct range restriction and measurement error. Note that a typographical error in Raju and Burke’s article has been corrected in this function so as to compute appropriate partial derivatives.

Usage

```r
estimate_var_tsa_meas(mean_rtp, var = 0, mean_qx = 1, mean_qy = 1, ...)
```

```r
estimate_var_tsa_uvdrr(
  mean_rtpa,
  var = 0,
  mean_ux = 1,
  mean_qxa = 1,
  mean_qyi = 1,
  ...
)
```

```r
estimate_var_tsa_bvdrr(
  mean_rtpa,
  var = 0,
  mean_ux = 1,
  mean_uy = 1,
  mean_qxa = 1,
  mean_qya = 1,
  ...
)
```

```r
estimate_var_tsa_uvirr(
  mean_rtpa,
  var = 0,
  mean_ut = 1,
  mean_qxa = 1,
  mean_qyi = 1,
  ...
)
```
estimate_var_tsa_bvirr(
    mean_rtpa,
    var = 0,
    mean_ux = 1,
    mean_uy = 1,
    mean_qxa = 1,
    mean_qya = 1,
    sign_rxz = 1,
    sign_ryz = 1,
    ...
)

estimate_var_tsa_rb1(
    mean_rtpa,
    var = 0,
    mean_ux = 1,
    mean_rxx = 1,
    mean_ryy = 1,
    ...
)

estimate_var_tsa_rb2(
    mean_rtpa,
    var = 0,
    mean_ux = 1,
    mean_qx = 1,
    mean_qy = 1,
    ...
)

Arguments

- **mean_rtp**  Mean corrected correlation.
- **var**  Variance to be corrected for artifacts.
- **mean_qx**  Mean square root of reliability for X.
- **mean_qy**  Mean square root of reliability for Y.
- **...**  Additional arguments.
- **mean_rtpa**  Mean corrected correlation.
- **mean_ux**  Mean observed-score u ratio for X.
- **mean_qxa**  Mean square root of unrestricted reliability for X.
- **mean_qyi**  Mean square root of restricted reliability for Y.
- **mean_uy**  Mean observed-score u ratio for Y.
- **mean_qya**  Mean square root of unrestricted reliability for Y.
- **mean_ut**  Mean true-score u ratio for X.
- **sign_rxz**  Sign of the relationship between X and the selection mechanism.
**estimate_var_tsa**

- **sign_ryz**: Sign of the relationship between Y and the selection mechanism.
- **mean_rxx**: Mean reliability for X.
- **mean_ryy**: Mean reliability for Y.

**Value**

Vector of variances corrected for mean artifacts via Taylor series approximation.

**Notes**

A typographical error in Raju and Burke’s article has been corrected in `estimate_var_tsa_rb2()` so as to compute appropriate partial derivatives.

**References**


**Examples**

```r
estimate_var_tsa_meas(mean_rtp = .5, var = .02,
                      mean_qx = .8,
                      mean_qy = .8)
estimate_var_tsa_uvdrr(mean_rtpa = .5, var = .02,
                      mean_ux = .8,
                      mean_qxa = .8,
                      mean_qyi = .8)
estimate_var_tsa_bvdrr(mean_rtpa = .5, var = .02,
                      mean_ux = .8,
                      mean_qx = .8,
                      mean_qxa = .8,
                      mean_qya = .8)
estimate_var_tsa_uvirr(mean_rtp = .5, var = .02,
                      mean_ut = .8,
                      mean_qxa = .8,
                      mean_qyi = .8)
estimate_var_tsa_bvirr(mean_rtp = .5, var = .02,
                      mean_ux = .8,
                      mean_qx = .8,
                      mean_qxa = .8,
                      mean_qya = .8,
                      sign_rxz = 1, sign_ryz = 1)
estimate_var_tsa_rb1(mean_rtpa = .5, var = .02,
                      mean_ux = .8,
                      mean_qx = .8,
                      mean_qxa = .8,
                      mean_qya = .8,
                      mean_rxx = .8)
```
mean_rxx = .8,
mean_ryy = .8)

estimate_var_tsa_rb2(mean_rtpa = .5, var = .02,
mean_ux = .8,
mean_qx = .8,
mean_qy = .8)

filter_ma

filter meta-analyses

Description

Filter psychmeta meta-analysis objects based on specified criteria.

Usage

filter_ma(
ma_obj,
alyses = "all",
macth = c("all", "any"),
case_sensitive = TRUE,
...
)

filter_meta(
ma_obj,
alyses = "all",
macth = c("all", "any"),
case_sensitive = TRUE,
...
)

Arguments

ma_obj A psychmeta meta-analysis object.
alyses Which analyses to extract? Can be either "all" to extract all meta-analyses in the object (default) or a list containing one or more of the following arguments:

• construct: A list or vector of construct names to search for.
• construct_pair: A list of vectors of construct pairs to search for. (e.g., list(c("X", "Y"), c("X", "Z"))).
• pair_id: A list or vector of numeric construct pair IDs (unique construct-pair indices).
• analysis_id: A list or vector of numeric analysis IDs (unique analysis indexes).
• k_min: A numeric value specifying the minimum k for extracted meta-analyses.
- **N_min**: A numeric value specifying the minimum N for extracted meta-analyses.

**match**

Should extracted meta-analyses match all (default) or any of the criteria given in analyses?

**case_sensitive**

Logical scalar that determines whether character values supplied in analyses should be treated as case sensitive (TRUE, default) or not (FALSE).

... Additional arguments.

**Value**

A psychmeta meta-analysis object with analyses matching the specified criteria.

**Examples**

```r
ma_obj <- ma_r(ma_method = "ic", rxyi = rxyi, n = n, rxx = rxxi, ryy = ryyi,
               construct_x = x_name, construct_y = y_name, sample_id = sample_id, citekey = NULL,
               moderators = moderator, data = data_r_meas_multi,
               impute_artifacts = FALSE, clean_artifacts = FALSE)
ma_obj <- ma_r_ad(ma_obj, correct_rr_x = FALSE, correct_rr_y = FALSE)
filter_ma(ma_obj, analyses="all")
filter_ma(ma_obj, analyses=list(construct="X"), match="all")
filter_ma(ma_obj, analyses=list(construct="X", k_min=21), match="any")
filter_ma(ma_obj, analyses=list(construct="X", k_min=21), match="all")
```

**Description**

A function to format numbers and logical values as characters for display purposes. Includes control over formatting of decimal digits, leading zeros, sign characters, and characters to replace logical, NA, NaN, and Inf values. Factors are converted to strings. Strings are returned verbatim.

**Usage**

```r
format_num(x, digits = 2L, decimal.mark = getOption("OutDec"),
           leading0 = "conditional", drop0integer = FALSE,
           neg.sign = "\u2212", pos.sign = "figure",
           big.mark = "\u202F", big.interval = 3L,
           small.mark = "\u202F", small.interval = 3L,
           na.mark = "\u2014", lgl.mark = c("+", "\u221E")
```
**Arguments**

- **x**: A vector, matrix, or data.frame of numbers to format
- **digits**: The number of decimal digits desired (used strictly; default: 2)
- **decimal.mark**: The character to use for the decimal point (defaults to locale default: `getOption("OutDec")`)
- **leading0**: How to print leading zeros on decimals. Can be logical to print (TRUE) or suppress (FALSE) leading zeros or a character string to substitute for leading zeros. If "conditional" (default), leading zeros are shown if a column contains any absolute values greater than 1 and suppressed otherwise. If "figure", leading zeros are replaced with a figure space (U+2007) if a column contains any absolute values greater than 1 and suppressed otherwise. If "figure_html", the same as "figure", but using the HTML entity for figure space (useful for Windows users in some locales).
- **drop0integer**: Logical. Should trailing decimal zeros be dropped for integers?
- **neg.sign**: Character to use as negative sign. Defaults to minus-sign (U+2212).
- **pos.sign**: Character to use as positive sign. Set to FALSE to suppress. If "figure" (default), the positive sign is a figure-space (U+2007) if a column contains any negative numbers and suppressed otherwise. If "figure_html", the same as "figure", but using the HTML entity for figure space (useful for Windows users in some locales).
- **big.mark**: Character to mark between each big.interval digits before the decimal point. Set to FALSE to suppress. Defaults to the SI/ISO 31-0 standard-recommended thin-spaces (U+202F).
- **big.interval**: See big.mark above; defaults to 3.
- **small.mark**: Character to mark between each small.interval digits after the decimal point. Set to FALSE to suppress. Defaults to the SI/ISO 31-0 standard-recommended thin-spaces (U+202F).
- **small.interval**: See small.mark above; defaults to 3.
- **na.mark**: Character to replace NA and NaN values. Defaults to em-dash (U+2014))
- **lgl.mark**: A length 2 vector containing characters to replace TRUE and FALSE. Defaults to c("+", "U+2212").
- **inf.mark**: A length 2 vector containing characters to replace Inf and -Inf. Defaults to c("+U+221e", "U+2212U+221e").

**Examples**

```r
# format_num() converts numeric values to characters with the specified formatting options.
# By default, thousands digit groups are separated by thin spaces, negative signs are replaced
# with minus signs, and positive signs and leading zeros are replaced with figure spaces
# (which have the same width as numbers and minus signs). These options ensure that all
# results will align neatly in columns when tabled.
format_num(x = c(10000, 1000, 2.41, -1.20, 0.41, -0.20))

# By default, format_num() uses your computer locale's default decimal mark as
# the decimal point. To force the usage of "." instead (e.g., for submission to
# a U.S. journal), set decimal.mark = ".":
```

```r
```
generate_bib

Generate a list of references included in meta-analyses

Description

This function generates a list of studies contributing to a meta-analysis

Usage

```r
generate_bib(
  ma_obj = NULL,
  bib = NULL,
  title.bib = NULL,
  style = "apa",
  additional_citekeys = NULL,
  file = NULL,
  output_dir = getwd(),
  output_format = c("word", "html", "pdf", "text", "odt", "rmd", "biblatex",
                   "citekeys"),
  analyses = "all",
  match = c("all", "any"),
  case_sensitive = TRUE,
  save_build_files = FALSE,
  header = list(),
  ...
)
```

Arguments

- `ma_obj`: A psychmeta meta-analysis object with citekeys supplied.
- `bib`: A BibTeX file containing the citekeys for the meta-analyses.
- `title.bib`: The title to give to the bibliography. If NULL, defaults to "Sources Contributing to Meta-Analyses".
- `style`: What style should references be formatted in? Can be a file path or URL for a CSL citation style or the style ID for any style available from the Zotero Style Repository. Defaults to APA style. (Retrieving a style by ID requires an internet connection. If unavailable, references will be rendered in Chicago style.).
generate_bib

additional_citekeys
   Additional citekeys to include in the reference list.

file
   The filename or filepath for the output file. If NULL, function will output directly
to the R console (if output_format is "text", a tibble with basic citation in-
formation; if "citekeys", the citekeys for included sources; otherwise, code to
generate the bibliography in an RMarkdown document).

output_dir
   The filepath for the output file. Defaults to the current working directory.

output_format
   The format of the output reference list. Available options are Word (default),
   HTML, PDF (requires LaTeX to be installed), ODT, or Rmarkdown, plain text,
   and BibLaTeX. Returning only the item citekeys is also possible. You can also
   specify the full name of another RMarkdown output_format.

analyses
   Which analyses to extract references for? See filter_ma() for details.

match
   Match "all" or "any" of the filter criteria? See filter_ma() for details.

case_sensitive
   Logical scalar that determines whether character values supplied in analyses
   should be treated as case sensitive (TRUE, default) or not (FALSE).

save_build_files
   Should the BibTeX and RMarkdown files used to generate the bibliography be
   saved (default: FALSE; always TRUE if file is NULL)?

header
   A list of YAML header parameters to pass to rmarkdown::render().

...Additional arguments to pass to rmarkdown::render().

Value

A list containing a tibble of bibtex reference data. Additionally, a reference list formatted in
the requested style and output_format is exported (or printed if file is "console").

See Also

Other output functions: metabulate_rmd_helper(), metabulate()

Examples

## Not run:
## Run a meta-analysis using ma_r() and include a citekey argument to provide
citation information for each source contributing to the meta-analyses.
ma_obj <- ma_r(ma_method = "ic", rxyi = rxyi, n = n, rxx = rxxi, ryy = ryyi,
   construct_x = x_name, construct_y = y_name, sample_id = sample_id,
   moderators = moderator, citekey = citekey, data = data_r_meas_multi)

## Next, use generate_bib() to generate the bibliography for the retained studies.
## The bib argument is the BibTeX or BibLaTeX .bib file containing the full
## reference information for each of the citekeys included in the meta-analysis database.
generate_bib(ma_obj, bib = system.file("templates/sample_bibliography.bib", package="psychmeta),
   file = "sample bibliography", output_dir = tempdir(), output_format = "word")

## End(Not run)
**generate_directory**  
*Generate a system of folders from a file path to a new directory*

**Description**
This function is intended to be helpful in simulations when directories need to be created and named according to values that are used or created within the simulation.

**Usage**
```r
genenerate_directory(path)
```

**Arguments**
- **path**  
The path to the directory to be created

**Value**
Creates a system of folders to a new directory.

---

**get_stuff**  
*Extract results from a psychmeta meta-analysis object*

**Description**
Functions to extract specific results from a meta-analysis tibble. This family of functions harvests information from meta-analysis objects and returns it as lists or tibbles that are easily navigable.

Available functions include:

- **get stuff**  
  Wrapper function for all other "get_" functions.
- **get_metatab**  
  Retrieve list of meta-analytic tables.
- **get_ad**  
  Retrieve list of artifact-distribution objects or a summary table of artifact descriptive statistics.
- **get_plots**  
  Retrieve list of meta-analytic plots.
- **get_escalc**  
  Retrieve list of escalc objects (i.e., effect-size data) for use with `metafor`.
- **get_metafor**  
  Alias for `get_escalc`.
- **get_followup**  
  Retrieve list of follow-up analyses.
• `get_leavelout`
  Retrieve list of leave-one-out meta-analyses (special case of `get_followup`).

• `get_cumulative`
  Retrieve list of cumulative meta-analyses (special case of `get_followup`).

• `get_bootstrap`
  Retrieve list of bootstrap meta-analyses (special case of `get_followup`).

• `get_metareg`
  Retrieve list of meta-regression analyses (special case of `get_followup`).

• `get_heterogeneity`
  Retrieve list of heterogeneity analyses (special case of `get_followup`).

• `get_matrix`
  Retrieve a tibble of matrices summarizing the relationships among constructs (only applicable to meta-analyses with multiple constructs).

**Usage**

```r
get_stuff(
  ma_obj,
  what = c("metatab", "escalc", "metafor", "ad", "followup", "heterogeneity", 
            "leavelout", "cumulative", "bootstrap", "metareg", "matrix", "plots"),
  analyses = "all",
  match = c("all", "any"),
  case_sensitive = TRUE,
  ma_method = c("bb", "ic", "ad"),
  correction_type = c("ts", "vgx", "vgy"),
  moderators = FALSE,
  as_ad_obj = TRUE,
  inputs_only = FALSE,
  ad_type = c("tsa", "int"),
  follow_up = c("heterogeneity", "leavelout", "cumulative", "bootstrap", "metareg"),
  plot_types = c("funnel", "forest", "leavelout", "cumulative"),
  ...
)
```

```r
get_escalc(
  ma_obj,
  analyses = "all",
  match = c("all", "any"),
  case_sensitive = TRUE,
  moderators = TRUE,
  ...
)
```

```r
get_metafor(
  ma_obj,
  analyses = "all",
  match = c("all", "any"),
  ...
)
```
get_metatab(
    ma_obj,
    analyses = "all",
    match = c("all", "any"),
    case_sensitive = TRUE,
    ma_method = c("bb", "ic", "ad"),
    correction_type = c("ts", "vgx", "vgy"),
    ...
)

get_ad(
    ma_obj,
    analyses = "all",
    match = c("all", "any"),
    case_sensitive = TRUE,
    ma_method = c("ad", "ic"),
    ad_type = c("tsa", "int"),
    as_ad_obj = FALSE,
    inputs_only = FALSE,
    ...
)

get_followup(
    ma_obj,
    analyses = "all",
    match = c("all", "any"),
    case_sensitive = TRUE,
    follow_up = c("heterogeneity", "leave1out", "cumulative", "bootstrap", "metareg"),
    ...
)

get_heterogeneity(
    ma_obj,
    analyses = "all",
    match = c("all", "any"),
    case_sensitive = TRUE,
    ...
)

get_leave1out(
    ma_obj,
    analyses = "all",
    match = c("all", "any"),
    ...
get_cumulative(
    ma_obj,
    analyses = "all",
    match = c("all", "any"),
    case_sensitive = TRUE,
    ...
)

get_bootstrap(
    ma_obj,
    analyses = "all",
    match = c("all", "any"),
    case_sensitive = TRUE,
    ...
)

get_metareg(
    ma_obj,
    analyses = "all",
    match = c("all", "any"),
    case_sensitive = TRUE,
    ...
)

get_matrix(
    ma_obj,
    analyses = "all",
    match = c("all", "any"),
    case_sensitive = TRUE,
    ...
)

get_plots(
    ma_obj,
    analyses = "all",
    match = c("all", "any"),
    case_sensitive = TRUE,
    plot_types = c("funnel", "forest", "leave1out", "cumulative"),
    ...
)

Arguments

ma_obj A psychmeta meta-analysis object.
For the `get_stuff()` function only: Character scalar telling `get_stuff()` what to get. All suffixes from functions in the "get_" family can be passed as arguments to `what`: "metatab", "escale", "metafor", "ad", "followup", "heterogeneity", "leave1out", "cumulative", "bootstrap", "metareg", "matrix", "plots"

analyses Which analyses to extract? Can be either "all" to extract references for all meta-analyses in the object (default) or a list containing one or more of the following arguments:

- `construct`: A list or vector of construct names to search for.
- `construct_pair`: A list of vectors of construct pairs to search for.
  (e.g., `list(c("X", "Y"), c("X", "Z"))`).
- `pair_id`: A list or vector of numeric construct pair IDs (unique construct-pair indices).
- `analysis_id`: A list or vector of numeric analysis IDs (unique analysis indexes).
- `k_min`: A numeric value specifying the minimum k for extracted meta-analyses.
- `N_min`: A numeric value specifying the minimum N for extracted meta-analyses.

match Should extracted meta-analyses match all (default) or any of the criteria given in analyses?

case_sensitive Logical scalar that determines whether character values supplied in analyses should be treated as case sensitive (TRUE, default) or not (FALSE).

ma_method Meta-analytic methods to be included. Valid options are: "bb", "ic", and "ad"

correction_type Types of meta-analytic corrections to be included. Valid options are: "ts", "vgx", and "vgy"

moderators Logical scalar that determines whether moderator variables should be included in escalc objects (TRUE; default) or not (FALSE).

as_ad_obj Logical scalar that determines whether artifact information should be returned as artifact-distribution objects (TRUE) or a summary table of artifact-distribution descriptive statistics (FALSE; default).

inputs_only Used only if `as_ad_obj = TRUE`: Logical scalar that determines whether artifact information should be returned as summaries of the raw input values (TRUE) or artifact values that may have been cross-corrected for range restriction and measurement error (FALSE; default).

ad_type Used only if `ma_method = "ic"`: Character value(s) indicating whether Taylor-series approximation artifact distributions ("tsa") and/or interactive artifact distributions ("int") should be retrieved.

follow_up Vector of follow-up analysis names (options are: "heterogeneity", "leave1out", "cumulative", "bootstrap", "metareg").

plot_types Vector of plot types (options are: "funnel", "forest", "leave1out", "cumulative"; multiple allowed).

... Additional arguments.
Value

Selected set of results.

Examples

```r
## Not run:
## Run meta-analysis:
ma_obj <- ma_r(ma_method = "ic", rxyi = rxyi, n = n, rxx = rxxi, ryy = ryyi,
             construct_x = x_name, construct_y = y_name,
             sample_id = sample_id, citekey = NULL,
             moderators = moderator, data = data_r_meas_multi,
             impute_artifacts = FALSE, clean_artifacts = FALSE)
ma_obj <- ma_r_ad(ma_obj, correct_rr_x = FALSE, correct_rr_y = FALSE)
## Run additional analyses:
ma_obj <- heterogeneity(ma_obj)
ma_obj <- sensitivity(ma_obj, boot_iter = 10, boot_ci_type = "norm")
ma_obj <- metareg(ma_obj)
ma_obj <- plot_funnel(ma_obj)
ma_obj <- plot_forest(ma_obj)
## View summary:
summary(ma_obj)
## Extract selected analyses:
get_metatab(ma_obj)
get_matrix(ma_obj)
get_escalc(ma_obj)
get_bootstrap(ma_obj)
get_cumulative(ma_obj)
get_leave1out(ma_obj)
get_heterogeneity(ma_obj)
get_metareg(ma_obj)
get_plots(ma_obj)
get_ad(ma_obj, ma_method = "ic", as_ad_obj = TRUE)
get_ad(ma_obj, ma_method = "ic", as_ad_obj = FALSE)
## Same extractions as above, but using get_stuff() and the "what" argument:
get_stuff(ma_obj, what = "metatab")
get_stuff(ma_obj, what = "matrix")
get_stuff(ma_obj, what = "escalc")
get_stuff(ma_obj, what = "bootstrap")
get_stuff(ma_obj, what = "cumulative")
get_stuff(ma_obj, what = "leave1out")
get_stuff(ma_obj, what = "heterogeneity")
get_stuff(ma_obj, what = "metareg")
get_stuff(ma_obj, what = "plots")
get_stuff(ma_obj, what = "ad", ma_method = "ic", as_ad_obj = TRUE)
get_stuff(ma_obj, what = "ad", ma_method = "ic", as_ad_obj = FALSE)
## End(Not run)
```
heterogeneity

Supplemental heterogeneity statistics for meta-analyses

Description

This function computes a variety of supplemental statistics for meta-analyses. The statistics here are included for interested users. It is strongly recommended that heterogeneity in meta-analysis be interpreted using the $SD_{res}$, $SD_{ρ}$, and $SD_{δ}$ statistics, along with corresponding credibility intervals, which are reported in the default ma_obj output (Wiernik et al., 2017).

Usage

heterogeneity(
  ma_obj,
  es_failsafe = NULL,
  conf_level = attributes(ma_obj)$inputs$conf_level,
  var_res_ci_method = c("profile_var_es", "profile_Q", "normal_logQ"),
  ...
)

Arguments

ma_obj | Meta-analysis object.

es_failsafe | Failsafe effect-size value for file-drawer analyses.

conf_level | Confidence level to define the width of confidence intervals (default is conf_level specified in ma_obj).

var_res_ci_method | Which method to use to estimate the limits. Options are profile_var_es for a profile-likelihood interval assuming $σ^2_{es} χ^2(k − 1)$, profile_Q for a profile-likelihood interval assuming $Q χ^2(k−1, λ)$, $λ = \sum_{i=1}^{k} w_i(θ − \bar{θ})^2$, and normal_logQ for a delta method assuming log(Q) follows a standard normal distribution.

... | Additional arguments.

Value

ma_obj with heterogeneity statistics added. Included statistics include:

es_type | The effect size metric used.

percent_var_accounted | Percent variance accounted for statistics (by sampling error, by other artifacts, and total). These statistics are widely reported, but not recommended, as they tend to be misinterpreted as suggesting only a small portion of the observed variance is accounted for by sampling error and other artifacts (Schmidt, 2010; Schmidt & Hunter, 2015, p. 15, 425). The square roots of these values are more interpretable and appropriate indices of the relations between observed effect sizes and statistical artifacts (see cor(es, perturbations)).
The correlation between observed effect sizes and statistical artifacts in each sample (with sampling error, with other artifacts, and with artifacts in total), computed as $\sqrt{\text{percent var accounted}}$. These indices are more interpretable and appropriate indices of the relations between observed effect sizes and statistical artifacts than percent_var_accounted.

$\text{rel_es_obs} = 1 - \frac{\text{var}_{\text{post}}}{\text{var}_{\text{pre}}}$, the reliability of observed effect size differences as indicators of true effect sizes differences in the sampled studies. This value is useful for correcting correlations between moderators and effect sizes in meta-regression.

$\text{H}_\text{squared}$ The ratio of the observed effect size variance to the predicted (error) variance. Also the square root of Q divided by its degrees of freedom.

$H$ The ratio of the observed effect size standard deviation to the predicted (error) standard deviation.

$I_\text{ squared}$ The estimated percent variance not accounted for by sampling error or other artifacts (attributable to moderators and uncorrected artifacts). This statistic is simply $\text{rel_es_obs}$ expressed as a percentage rather than a decimal.

$Q$ Cochran’s $\chi^2$ statistic. Significance tests using this statistic are strongly discouraged; heterogeneity should instead be determined by examining the width of the credibility interval and the practical differences between effect sizes contained within it (Wiernik et al., 2017). This value is not accurate when artifact distribution methods are used for corrections.

$\tau_\text{squared}$, an estimator of the random effects variance component (analogous to the Hunter-Schmidt $SD_{res}^2$, $SD_{\rho}^2$, or $SD_{\delta}^2$ statistics), with its confidence interval. This value is not accurate when artifact distribution methods are used for corrections.

$\tau$, $\sqrt{\tau_\text{squared}}$, analogous to the Hunter-Schmidt $SD_{res}^2$, $SD_{\rho}^2$, and $SD_{\delta}^2$ statistics, with its confidence interval. This value is not accurate when artifact distribution methods are used for corrections.

$Q_r$, $H_r_\text{squared}$, $H_r$, $I_r_\text{squared}$, $\tau_r_\text{squared}$, $\tau_r$ Outlier-robust versions of these statistics, computed based on absolute deviations from the weighted mean effect size (see Lin et al., 2017). These values are not accurate when artifact distribution methods are used for corrections.

$Q_m$, $H_m_\text{squared}$, $H_m$, $I_m_\text{squared}$, $\tau_m_\text{squared}$, $\tau_m$ Outlier-robust versions of these statistics, computed based on absolute deviations from the weighted median effect size (see Lin et al., 2017). These values are not accurate when artifact distribution methods are used for corrections.

$file\_drawer$ Fail-safe $N$ and $k$ statistics (file-drawer analyses). These statistics should not be used to evaluate publication bias, as they counterintuitively suggest less when publication bias is strong (Becker, 2005). However, in the absence of publication bias, they can be used as an index of second-order sampling error (how likely is a mean effect to reduce to the specified value with additional studies?). The confidence interval around the mean effect can be used more directly for the same purpose.

Results are reported using computation methods described by Schmidt and Hunter. For bare-bones and individual-correction meta-analyses, results are also reported using computation methods described by DerSimonian and Laird, outlier-robust computation methods, and, if weights from metafor are used, heterogeneity results from metafor.
limits\_tau

References


Examples

```r
## Correlations
ma_obj <- ma_r_ic(rxyi = rxyi, n = n, rxx = rxxi, ryy = ryyi, ux = ux, correct_rr_y = FALSE, data = data_r_uvirr)
ma_obj <- ma_r_ad(ma_obj, correct_rr_y = FALSE)
ma_obj <- heterogeneity(ma_obj = ma_obj)
ma_obj$heterogeneity[[1]]$barebones
ma_obj$heterogeneity[[1]]$individual_correction$true_score
ma_obj$heterogeneity[[1]]$artifact_distribution$true_score

## d values
ma_obj <- ma_d_ic(d = d, n1 = n1, n2 = n2, ryy = ryyi, data = data_d_meas_multi)
ma_obj <- ma_d_ad(ma_obj)
ma_obj <- heterogeneity(ma_obj = ma_obj)
ma_obj$heterogeneity[[1]]$barebones
ma_obj$heterogeneity[[1]]$individual_correction$latentGroup_latentY
ma_obj$heterogeneity[[1]]$artifact_distribution$latentGroup_latentY
```

<table>
<thead>
<tr>
<th>limits_tau</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Confidence limits of tau</em></td>
</tr>
</tbody>
</table>

Description
Note that this interval does not incorporate uncertainty in artifact estimates, so the interval will be somewhat conservative when applied to individual-correction or artifact-distribution meta-analyses.
Usage

```r
limits_tau(
  var_es,
  var_pre,
  k,
  method = c("profile_var_es", "profile_Q", "normal_logQ"),
  conf_level = 0.95,
  var_unbiased = TRUE
)
```

Arguments

- `var_es`: The observed variance of effect sizes.
- `var_pre`: The predicted variance of effect sizes due to artifacts.
- `k`: The number of studies in a meta-analysis.
- `method`: Which method to use to estimate the limits. Options are `profile_var_es` for a profile-likelihood interval assuming $\sigma^2 e \chi^2(k - 1)$, `profile_Q` for a profile-likelihood interval assuming $Q \chi^2(k - 1, \lambda)$, $\lambda = \sum_{i=1} k w_i (\theta - \bar{\theta})^2$, and `normal_logQ` for a delta method assuming log(Q) follows a standard normal distribution.
- `conf_level`: Confidence level.
- `var_unbiased`: Are variances computed using the unbiased (TRUE) or maximum likelihood (FALSE) estimator?

Value

The confidence limits of tau

Examples

```r
limits_tau(var_es = 0.008372902, var_pre = 0.004778935, k = 20)
```

Description

Note that this interval does not incorporate uncertainty in artifact estimates, so the interval will be somewhat conservative when applied to individual-correction or artifact-distribution meta-analyses.
Usage

limits_tau2(
  var_es,
  var_pre,
  k,
  method = c("profile_var_es", "profile_Q", "normal_logQ"),
  conf_level = 0.95,
  var_unbiased = TRUE
)

Arguments

var_es The observed variance of effect sizes.
var_pre The predicted variance of effect sizes due to artifacts.
k The number of studies in a meta-analysis.
method Which method to use to estimate the limits. Options are profile_var_es for a profile-likelihood interval assuming $\sigma^2$ $\chi^2(k - 1)$, profile_Q for a profile-likelihood interval assuming $Q$ $\chi^2(k - 1, \lambda)$, $\lambda = \sum_{i=1}^k w_i (\theta - \bar{\theta})^2$, and normal_logQ for a delta method assuming log(Q) follows a standard normal distribution.
conf_level Confidence level.
var_unbiased Are variances computed using the unbiased (TRUE) or maximum likelihood (FALSE) estimator?

Value

The confidence limits of tau-squared

Examples

limits_tau2(var_es = 0.008372902, var_pre = 0.004778935, k = 20)

---

`lm_mat`  

*Compute linear regression models and generate "lm" objects from covariance matrices.*

**Description**

Compute linear regression models and generate "lm" objects from covariance matrices.
Usage

lm_mat(
  formula,
  cov_mat,
  mean_vec = rep(0, ncol(cov_mat)),
  n = Inf,
  se_beta_method = c("lm", "normal"),
  ...
)

matrixreg(
  formula,
  cov_mat,
  mean_vec = rep(0, ncol(cov_mat)),
  n = Inf,
  se_beta_method = c("lm", "normal"),
  ...
)

matreg(
  formula,
  cov_mat,
  mean_vec = rep(0, ncol(cov_mat)),
  n = Inf,
  se_beta_method = c("lm", "normal"),
  ...
)

lm_matrix(
  formula,
  cov_mat,
  mean_vec = rep(0, ncol(cov_mat)),
  n = Inf,
  se_beta_method = c("lm", "normal"),
  ...
)

Arguments

formula Regression formula with a single outcome variable on the left-hand side and one or more predictor variables on the right-hand side (e.g., Y ~ X1 + X2).
cov_mat Covariance matrix containing the variables to be used in the regression.
mean_vec Vector of means corresponding to the variables in cov_mat.
n Sample size to be used in significance testing
se_beta_method Method to use to estimate the standard errors of standardized regression (beta) coefficients. Current options include "lm" (estimate standard errors using conventional regression formulas) and "normal" (use the Jones-Waller normal-theory
approach from the fungible::seBeta() and fungible::seBetaCor() functions

... Additional arguments.

Value

An object with the class "lm_mat" that can be used with summary, print, predict, and anova methods.

Examples

## Generate data
S <- reshape_vec2mat(cov = c(.3 * 2 * 3,
     .4 * 2 * 4,
     .5 * 3 * 4),
     var = c(2, 3, 4)^2,
     var_names = c("X", "Y", "Z"))
mean_vec <- setNames(c(1, 2, 3), colnames(S))
dat <- data.frame(MASS::mvrnorm(n = 100, mu = mean_vec,
     Sigma = S, empirical = TRUE))

## Compute regression models with lm
lm_out1 <- lm(Y ~ X, data = dat)
lm_out2 <- lm(Y ~ X + Z, data = dat)

## Compute regression models with lm_mat
matreg_out1 <- lm_mat(formula = Y ~ X, cov_mat = S, mean_vec = mean_vec, n = nrow(dat))
matreg_out2 <- lm_mat(formula = Y ~ X + Z, cov_mat = S, mean_vec = mean_vec, n = nrow(dat))

## Compare results of lm and lm_mat with one predictor
lm_out1
matreg_out1

## Compare summaries of lm and lm_mat with one predictor
summary(lm_out1)
summary(matreg_out1)

## Compare results of lm and lm_mat with two predictors
lm_out2
matreg_out2

## Compare summaries of lm and lm_mat with two predictors
summary(lm_out2)
summary(matreg_out2)

## Compare predictions made with lm and lm_mat
predict(object = matreg_out1, newdata = data.frame(X = 1:5))
predict(object = summary(matreg_out1), newdata = data.frame(X = 1:5))
predict(lm_out1, newdata = data.frame(X = 1:5))

## Compare predictions made with lm and lm_mat (with confidence intervals)
predict(object = matreg_out1, newdata = data.frame(X = 1:5),
     se.fit = TRUE, interval = "confidence")
predict(lm_out1, newdata = data.frame(X = 1:5),
          se.fit = TRUE, interval = "confidence")

## Compare predictions made with lm and lm_mat (with prediction intervals)
predict(object = matreg_out1, newdata = data.frame(X = 1:5),
          se.fit = TRUE, interval = "prediction")
predict(lm_out1, newdata = data.frame(X = 1:5),
          se.fit = TRUE, interval = "prediction")

## Compare model comparisons computed using lm and lm_mat objects
anova(lm_out1, lm_out2)
anova(matreg_out1, matreg_out2)

## Model comparisons can be run on lm_mat summaries, too:
anova(summary(matreg_out1), summary(matreg_out2))
## Or summaries and raw models can be mixed:
anova(matreg_out1, summary(matreg_out2))
anova(summary(matreg_out1), matreg_out2)

## Compare confidence intervals computed using lm and lm_mat objects
confint(object = lm_out1)
confint(object = matreg_out1)
confint(object = summary(matreg_out1))

confint(object = lm_out2)
confint(object = matreg_out2)
confint(object = summary(matreg_out2))

---

**ma_d**

Meta-analysis of d values

**Description**

The `ma_r_bb`, `ma_r_ic`, and `ma_r_ad` functions implement bare-bones, individual-correction, and artifact-distribution correction methods for d values, respectively. The `ma_d` function is the master function for meta-analyses of d values - it facilitates the computation of bare-bones, artifact-distribution, and individual-correction meta-analyses of correlations for any number of group-wise contrasts and any number of dependent variables. When artifact-distribution meta-analyses are performed, `ma_d` will automatically extract the artifact information from a database and organize it into the requested type of artifact distribution object (i.e., either Taylor series or interactive artifact distributions). `ma_d` is also equipped with the capability to clean databases containing inconsistently recorded artifact data, impute missing artifacts (when individual-correction meta-analyses are requested), and remove dependency among samples by forming composites or averaging effect sizes and artifacts. The automatic compositing features in `ma_d` are employed when `sample_ids` and/or construct names are provided.
Usage

ma_d(
  d,
  nl,
  n2 = NULL,
  n_adj = NULL,
  sample_id = NULL,
  citekey = NULL,
  treat.as.r = FALSE,
  ma_method = c("bb", "ic", "ad"),
  ad_type = c("tsa", "int"),
  correction_method = "auto",
  group_id = NULL,
  group1 = NULL,
  group2 = NULL,
  group_order = NULL,
  construct_y = NULL,
  facet_y = NULL,
  measure_y = NULL,
  construct_order = NULL,
  wt_type = c("n_effective", "sample_size", "inv_var_mean", "inv_var_sample", "DL",
              "HE", "HS", "SJ", "ML", "REML", "EB", "PM"),
  correct_bias = TRUE,
  correct_rel = NULL,
  correct_rGg = FALSE,
  correct_ryy = TRUE,
  correct_rr = NULL,
  correct_rr_g = TRUE,
  indirect_rr = NULL,
  indirect_rr_g = TRUE,
  indirect_rr_y = TRUE,
  rGg = NULL,
  pi = NULL,
  pa = NULL,
  ryy = NULL,
  ryy_restricted = TRUE,
  ryy_type = "alpha",
  k_items_y = NULL,
  uy = NULL,
  uy_observed = TRUE,
  sign_rz = NULL,
  sign_rgz = 1,
  sign_ryz = 1,
  moderators = NULL,
  cat_moderators = TRUE,
  moderator_type = c("simple", "hierarchical", "none"),
  supplemental_ads = NULL,
data = NULL,
control = control_psychmeta(),
...)

ma_d_ad(
  ma_obj,
ad_obj_g = NULL,
ad_obj_y = NULL,
correction_method = "auto",
use_ic_ads = c("tsa", "int"),
correct_rGg = FALSE,
correct_ryy = TRUE,
correct_rr_g = TRUE,
correct_rr_y = TRUE,
indirect_rr_g = TRUE,
indirect_rr_y = TRUE,
sign_rgz = 1,
sign_ryz = 1,
control = control_psychmeta(),
...)

ma_d_bb(
d,
n1,
n2 = rep(NA, length(d)),
n_adj = NULL,
sample_id = NULL,
citekey = NULL,
wt_type = c("n_effective", "sample_size", "inv_var_mean", "inv_var_sample", "DL",
    "HE", "HS", "SJ", "ML", "REML", "EB", "PM"),
correct_bias = TRUE,
moderators = NULL,
cat_moderators = TRUE,
moderator_type = c("simple", "hierarchical", "none"),
data = NULL,
control = control_psychmeta(),
...)

ma_d_ic(
d,
n1,
n2 = NULL,
n_adj = NULL,
sample_id = NULL,
citekey = NULL,
ma_d
treat_as_r = FALSE,
wt_type = c("n_effective", "sample_size", "inv_var_mean", "inv_var_sample", "DL", "HE", "HS", "SJ", "ML", "REML", "EB", "PM"),
correct_bias = TRUE,
correct_rGg = FALSE,
correct_ryy = TRUE,
correct_rr_g = FALSE,
correct_rr_y = TRUE,
indirect_rr_g = TRUE,
indirect_rr_y = TRUE,
rGg = NULL,
pi = NULL,
pa = NULL,
ryy = NULL,
ryy_restricted = TRUE,
ryy_type = "alpha",
k_items_y = NULL,
uy = NULL,
uy_observed = TRUE,
sign_rgz = 1,
sign_ryz = 1,
mmoderators = NULL,
cat_moderators = TRUE,
mmoderator_type = c("simple", "hierarchical", "none"),
supplemental_ads_y = NULL,
data = NULL,
control = control_psychmeta(),
...)

Arguments

d Vector or column name of observed d values. NOTE: Beginning in psychmeta version 2.5.2, d values of exactly 0 in individual-correction meta-analyses are replaced with a functionally equivalent value (in the correlation metric) via the zero_substitute argument for control_psychmeta to facilitate the estimation of corrected error variances.

n1 Vector or column name of sample sizes.
n2 Vector or column name of sample sizes.
n_adj Optional: Vector or column name of sample sizes adjusted for sporadic artifact corrections.
sample_id Optional vector of identification labels for samples/studies in the meta-analysis.
citekey Optional vector of bibliographic citation keys for samples/studies in the meta-analysis (if multiple citekeys pertain to a given effect size, combine them into a single string entry with comma delimiters (e.g., "citekey1,citekey2").
treat_as_r Logical scalar determining whether d values are to be meta-analyzed as d values (FALSE; default) or whether they should be meta-analyzed as correlations and have the final results converted to the d metric (TRUE).
ma_method
Method to be used to compute the meta-analysis: "bb" (barebones), "ic" (individual correction), or "ad" (artifact distribution).

ad_type
For when ma_method is "ad", specifies the type of artifact distribution to use: "int" or "tsa".

correction_method
Character scalar or a matrix with group_id levels as row names and construct_y levels as column names. When ma_method is "ad", select one of the following methods for correcting artifacts: "auto", "meas", "uvdrr", "uvirr", "bvdrr", "bvrr", "rbOrig", "rb1Orig", "rb2Orig", "rbAdj", "rb1Adj", and "rb2Adj". (note: "rb1Orig", "rb2Orig", "rb1Adj", and "rb2Adj" can only be used when Taylor series artifact distributions are provided and "rbOrig" and "rbAdj" can only be used when interactive artifact distributions are provided). See "Details" of ma_d_ad for descriptions of the available methods.

group_id
Vector of group comparison IDs (e.g., Treatment1-Control, Treatment2-Control). The group_id argument supersedes the group1 and group2 arguments. If group_id is not NULL, the values supplied to the group_order argument must correspond to group_id values.

group1, group2
Vector of group identification labels (e.g., Treatment1, Treatment2, Control)

group_order
Optional vector indicating the order in which (1) group1 and group2 values or (2) group_ids should be arranged. If group_order is NULL, the order of group pairings will be determined internally using alpha-numeric ordering.

construct_y
Vector of construct names for construct designated as "Y".

facet_y
Vector of facet names for constructs designated as "Y". Facet names "global", "overall", and "total" are reserved to indicate observations that represent effect sizes that have already been composited or that represent construct-level measurements rather than facet-level measurements. To avoid double-compositing, any observation with one of these reserved names will only be eligible for auto-compositing with other such observations and will not be combined with narrow facets.

measure_y
Vector of names for measures associated with constructs designated as "Y".

construct_order
Vector indicating the order in which Y variables should be arranged.

wt_type
Type of weight to use in the meta-analysis: options are "n_effective" (effective sample size), "sample_size", "inv_var_mean" (inverse variance computed using mean effect size), and "inv_var_sample" (inverse variance computed using sample-specific effect sizes). Supported options borrowed from metafor are "DL", "HE", "HS", "SJ", "ML", "REML", "EB", and "PM" (see metafor documentation for details about the metafor methods).

correct_bias
Logical scalar that determines whether to correct correlations for small-sample bias (TRUE) or not (FALSE).

correct_rel
Optional named vector that supersedes correct_rGg and correct_ryy. Names should correspond to construct names in group_id and construct_y to determine which constructs should be corrected for unreliability.

correct_rGg
Logical scalar or vector that determines whether to correct the grouping variable variable for measurement error (TRUE) or not (FALSE).
**correct_ryy**  Logical scalar or vector that determines whether to correct the Y variable for measurement error (TRUE) or not (FALSE).

**correct_rr**  Optional named vector that supersedes `correct_rr_g` and `correct_rr_y`. Names should correspond to construct names in `group_id` and `construct_y` to determine which constructs should be corrected for range restriction.

**correct_rr_g**  Logical scalar or vector or column name determining whether each d value should be corrected for range restriction in the grouping variable (TRUE) or not (FALSE).

**correct_rr_y**  Logical scalar or vector or column name determining whether each d should be corrected for range restriction in Y (TRUE) or not (FALSE).

**indirect_rr**  Optional named vector that supersedes `indirect_rr_g` and `indirect_rr_y`. Names should correspond to construct names in `group_id` and `construct_y` to determine which constructs should be corrected for indirect range restriction.

**indirect_rr_g**  Logical vector or column name determining whether each d should be corrected for indirect range restriction in the grouping variable (TRUE) or not (FALSE). Superseded in evaluation by `correct_rr_g` (i.e., if `correct_rr_g` == FALSE, the value supplied for `indirect_rr_g` is disregarded).

**indirect_rr_y**  Logical vector or column name determining whether each d should be corrected for indirect range restriction in Y (TRUE) or not (FALSE). Superseded in evaluation by `correct_rr_y` (i.e., if `correct_rr_y` == FALSE, the value supplied for `indirect_rr_y` is disregarded).

**rGg**  Vector or column name of reliability estimates for X.

**pi**  Scalar or vector containing the restricted-group proportions of group membership. If a vector, it must either (1) have as many elements as there are d values or (2) be named so as to match with levels of the `group_id` argument.

**pa**  Scalar or vector containing the unrestricted-group proportions of group membership (default = .5). If a vector, it must either (1) have as many elements as there are d values or (2) be named so as to match with levels of the `group_id` argument.

**ryy**  Vector or column name of reliability estimates for Y.

**ryy_restricted**  Logical vector or column name determining whether each element of `ryy` is an incumbent reliability (TRUE) or an applicant reliability (FALSE).

**ryy_type**  String vector identifying the types of reliability estimates supplied (e.g., "alpha", "retest", "interrater_r", "splithalf"). See the documentation for `ma_r` for a full list of acceptable reliability types.

**k_items_y**  Numeric vector identifying the number of items in each scale.

**uy**  Vector or column name of u ratios for Y.

**uy_observed**  Logical vector or column name determining whether each element of `uy` is an observed-score u ratio (TRUE) or a true-score u ratio (FALSE).

**sign_rz**  Optional named vector that supersedes `sign_rgz` and `sign_ryz`. Names should correspond to construct names in `group_id` and `construct_y` to determine the sign of each construct’s relationship with the selection mechanism.

**sign_rgz**  Sign of the relationship between X and the selection mechanism (for use with bvirr corrections only).
**sign_ryz**  
Sign of the relationship between Y and the selection mechanism (for use with bvrir corrections only).

**moderators**  
Matrix or column names of moderator variables to be used in the meta-analysis (can be a vector in the case of one moderator).

**cat_moderators**  
Logical scalar or vector identifying whether variables in the moderators argument are categorical variables (TRUE) or continuous variables (FALSE).

**moderator_type**  
Type of moderator analysis: "none" means that no moderators are to be used, "simple" means that moderators are to be examined one at a time, "hierarchical" means that all possible combinations and subsets of moderators are to be examined, and "all" means that simple and hierarchical moderator analyses are to be performed.

**supplemental_ads**  
Named list (named according to the constructs included in the meta-analysis) of supplemental artifact distribution information from studies not included in the meta-analysis. This is a list of lists, where the elements of a list associated with a construct are named like the arguments of the create_ad() function.

**data**  
Data frame containing columns whose names may be provided as arguments to vector arguments and/or moderators.

**control**  
Output from the control_psychmeta() function or a list of arguments controlled by the control_psychmeta() function. Ellipsis arguments will be screened for internal inclusion in control.

**...**  
Further arguments to be passed to functions called within the meta-analysis.

**ma_obj**  
For ma_d_ad only: Meta-analysis object of correlations or d values (regardless of input metric, output metric will be d).

**ad_obj_g**  
For ma_d_ad only: Artifact-distribution object for the grouping variable (output of the link{create_ad} or link{create_ad_group} functions). If ma_obj is of the class ma_master (i.e., the output of ma_r or ma_d), the object supplied for ad_obj_g must be a named list of artifact distributions with names corresponding to the "X" constructs in the meta-analyses contained within ma_obj.

**ad_obj_y**  
For ma_d_ad only: Artifact-distribution object for the Y variable (output of the create_ad function). If ma_obj is of the class ma_master, the object supplied for ad_obj_y must be a named list of artifact distributions with names corresponding to the "Y" constructs in the meta-analyses contained within ma_obj.

**use_ic_ads**  
For ma_d_ad only: Determines whether artifact distributions should be extracted from the individual correction results in ma_obj. Only evaluated when ad_obj_g or ad_obj_y is NULL and ma_obj does not contain individual correction results. Use one of the following commands: tsa to use the Taylor series method or int to use the interactive method.

**supplemental_ads_y**  
For ma_d_ic only: List supplemental artifact distribution information from studies not included in the meta-analysis. The elements of this list are named like the arguments of the create_ad() function.

---

**Details**

The options for correction_method are:
Automatic selection of the most appropriate correction procedure, based on the available artifacts and the logical arguments provided to the function. (default)

Correction for measurement error only.

Correction for univariate direct range restriction (i.e., Case II). The choice of which variable to correct for range restriction is made using the `correct_rr_x` and `correct_rr_y` arguments.

Correction for univariate indirect range restriction (i.e., Case IV). The choice of which variable to correct for range restriction is made using the `correct_rr_x` and `correct_rr_y` arguments.

Correction for bivariate direct range restriction. Use with caution: This correction is an approximation only and is known to have a positive bias.

Correction for bivariate indirect range restriction (i.e., Case V).

Not recommended: Raju and Burke’s version of the correction for direct range restriction, applied interactively. We recommend using "uvdrr" instead.

Not recommended: Raju and Burke’s version of the correction for direct range restriction, applied interactively. Adjusted to account for range restriction in the reliability of the Y variable. We recommend using "uvdrr" instead.

Not recommended: Raju and Burke’s version of the correction for direct range restriction, applied using their TSA1 method. We recommend using "uvdrr" instead.

Not recommended: Raju and Burke’s version of the correction for direct range restriction, applied using their TSA1 method. Adjusted to account for range restriction in the reliability of the Y variable. We recommend using "uvdrr" instead.

Not recommended: Raju and Burke’s version of the correction for direct range restriction, applied using their TSA2 method. We recommend using "uvdrr" instead.

Not recommended: Raju and Burke’s version of the correction for direct range restriction, applied using their TSA2 method. Adjusted to account for range restriction in the reliability of the Y variable. We recommend using "uvdrr" instead.

A nested tabular object of the class "ma_psychmeta". Components of output tables for bare-bones meta-analyses:

- `Pair_ID`
  Unique identification number for each construct-contrast pairing.
- **group_contrast**
  Name of the variable analyzed as the group-contrast variable.

- **construct_y**
  Name of the variable analyzed as construct Y.

- **analysis_id**
  Unique identification number for each analysis.

- **analysis_type**
  Type of moderator analyses: Overall, Simple Moderator, or Hierarchical Moderator.

- **k**
  Number of effect sizes meta-analyzed.

- **N**
  Total sample size of all effect sizes in the meta-analysis.

- **mean_d**
  Mean observed $d$ value.

- **var_d**
  Weighted variance of observed $d$ values.

- **var_e**
  Predicted sampling-error variance of observed $d$ values.

- **var_res**
  Variance of observed $d$ values after removing predicted sampling-error variance.

- **sd_d**
  Square root of $\text{var}_r$.

- **se_d**
  Standard error of mean_d.

- **sd_e**
  Square root of $\text{var}_e$.

- **sd_res**
  Square root of $\text{var}_\text{res}$.

- **CI_LL_XX**
  Lower limit of the confidence interval around mean_d, where "XX" represents the confidence level as a percentage.

- **CI_UL_XX**
  Upper limit of the confidence interval around mean_d, where "XX" represents the confidence level as a percentage.

- **CR_LL_XX**
  Lower limit of the credibility interval around mean_d, where "XX" represents the credibility level as a percentage.

- **CR_UL_XX**
  Upper limit of the credibility interval around mean_d, where "XX" represents the credibility level as a percentage.

Components of output tables for individual-correction meta-analyses:

- **pair_id**
  Unique identification number for each construct-contrast pairing.
• **group_contrast**
  Name of the variable analyzed as the group-contrast variable.

• **construct_y**
  Name of the variable analyzed as construct Y.

• **analysis_id**
  Unique identification number for each analysis.

• **analysis_type**
  Type of moderator analyses: Overall, Simple Moderator, or Hierarchical Moderator.

• **k**
  Number of effect sizes meta-analyzed.

• **N**
  Total sample size of all effect sizes in the meta-analysis.

• **mean_d**
  Mean observed $d$ value.

• **var_d**
  Weighted variance of observed $d$ values.

• **var_e**
  Predicted sampling-error variance of observed $d$ values.

• **var_res**
  Variance of observed $d$ values after removing predicted sampling-error variance.

• **sd_d**
  Square root of var_r.

• **se_d**
  Standard error of mean_d.

• **sd_e**
  Square root of var_e.

• **sd_res**
  Square root of var_res.

• **mean_delta**
  Mean artifact-corrected $d$ value.

• **var_d_c**
  Variance of artifact-corrected $d$ values.

• **var_e_c**
  Predicted sampling-error variance of artifact-corrected $d$ values.

• **var_delta**
  Variance of artifact-corrected $d$ values after removing predicted sampling-error variance.

• **sd_d_c**
  Square root of var_r_c.

• **se_d_c**
  Standard error of mean_delta.

• **sd_e_c**
  Square root of var_e_c.
- **sd_delta**
  Square root of var_delta.
- **CI_LL_XX**
  Lower limit of the confidence interval around mean_delta, where "XX" represents the confidence level as a percentage.
- **CI_UL_XX**
  Upper limit of the confidence interval around mean_delta, where "XX" represents the confidence level as a percentage.
- **CR_LL_XX**
  Lower limit of the credibility interval around mean_delta, where "XX" represents the credibility level as a percentage.
- **CR_UL_XX**
  Upper limit of the credibility interval around mean_delta, where "XX" represents the credibility level as a percentage.

Components of output tables for artifact-distribution meta-analyses:

- **pair_id**
  Unique identification number for each construct-contrast pairing.
- **group_contrast**
  Name of the variable analyzed as the group-contrast variable.
- **construct_y**
  Name of the variable analyzed as construct Y.
- **analysis_id**
  Unique identification number for each analysis.
- **analysis_type**
  Type of moderator analyses: Overall, Simple Moderator, or Hierarchical Moderator.
- **k**
  Number of effect sizes meta-analyzed.
- **N**
  Total sample size of all effect sizes in the meta-analysis.
- **mean_d**
  Mean observed $d$ value.
- **var_d**
  Weighted variance of observed $d$ values.
- **var_e**
  Predicted sampling-error variance of observed $d$ values.
- **var_art**
  Amount of variance in observed $d$ values that is attributable to measurement-error and range-restriction artifacts.
- **var_pre**
  Total predicted artifactual variance (i.e., the sum of var_e and var_art).
- **var_res**
  Variance of observed $d$ values after removing predicted sampling-error variance and predicted artifact variance.
• sd_d
  Square root of var_d.
• se_d
  Standard error of mean_d.
• sd_e
  Square root of var_e.
• sd_art
  Square root of var_art.
• sd_pre
  Square root of var_pre.
• sd_res
  Square root of var_res.
• mean_delta
  Mean artifact-corrected d value.
• var_d
  Weighted variance of observed d values corrected to the metric of delta.
• var_e
  Predicted sampling-error variance of observed d values corrected to the metric of delta.
• var_art
  Amount of variance in observed d values that is attributable to measurement-error and range-restriction artifacts corrected to the metric of delta.
• var_pre
  Total predicted artifactual variance (i.e., the sum of var_e and var_art) corrected to the metric of delta.
• var_delta
  Variance of artifact-corrected d values after removing predicted sampling-error variance and predicted artifactual variance.
• sd_d
  Square root of var_d corrected to the metric of delta.
• se_d
  Standard error of mean_d corrected to the metric of delta.
• sd_e
  Square root of var_e corrected to the metric of delta.
• sd_art
  Square root of var_art corrected to the metric of delta.
• sd_pre
  Square root of var_pre corrected to the metric of delta.
• sd_delta
  Square root of var_delta.
• CI_LL_XX
  Lower limit of the confidence interval around mean_delta, where "XX" represents the confidence level as a percentage.
• **CI_UL_XX**  
  Upper limit of the confidence interval around mean_delta, where "XX" represents the confidence level as a percentage.

• **CR_LL_XX**  
  Lower limit of the credibility interval around mean_delta, where "XX" represents the credibility level as a percentage.

• **CR_UL_XX**  
  Upper limit of the credibility interval around mean_delta, where "XX" represents the credibility level as a percentage.

**Note**

The difference between "rb" methods with the "orig" and "adj" suffixes is that the original does not account for the impact of range restriction on criterion reliabilities, whereas the adjusted procedure attempts to estimate the applicant reliability information for the criterion. The "rb" procedures are included for posterity: We strongly recommend using the "uvdrr" procedure to appropriately correct for univariate range restriction.

**References**


**Examples**

```r
### Demonstration of ma_d ###
## The 'ma_d' function can compute multi-construct bare-bones meta-analyses:
ma_d(d = d, n1 = n1, n2 = n2, construct_y = construct, data = data_d_meas_multi)

## It can also perform multiple individual-correction meta-analyses:
ma_d(ma_method = "ic", d = d, n1 = n1, n2 = n2, ryy = ryyi, 
    construct_y = construct, data = data_d_meas_multi)

## And 'ma_d' can also curate artifact distributions and compute multiple
## artifact-distribution meta-analyses:
ma_d(ma_method = "ad", d = d, n1 = n1, n2 = n2, 
    ryy = ryyi, correct_rr_y = FALSE, 
    construct_y = construct, data = data_d_meas_multi)

### Demonstration of ma_d_bb ###
## Example meta-analyses using simulated data:
```
### Demonstration of ma_d_bb ###
#### Example meta-analyses using simulated data: ####

- `ma_d_bb(d = d, n1 = n1, n2 = n2, data = data_d_meas_multi[data_d_meas_multi$construct == "Y",])`
- `ma_d_bb(d = d, n1 = n1, n2 = n2, data = data_d_meas_multi[data_d_meas_multi$construct == "Z",])`

### Demonstration of ma_d_ic ###
#### Example meta-analyses using simulated data: ####

- `ma_d_ic(d = d, n1 = n1, n2 = n2, ryy = ryyi, correct_rr_y = FALSE, data = data_d_meas_multi[data_d_meas_multi$construct == "Y",])`
- `ma_d_ic(d = d, n1 = n1, n2 = n2, ryy = ryyi, correct_rr_y = FALSE, data = data_d_meas_multi[data_d_meas_multi$construct == "Z",])`

### Second-order meta-analysis function for d values

**Description**

This function computes second-order meta-analysis function for $d$ values. It supports second-order analyses of bare-bones, artifact-distribution, and individual-correction meta-analyses.

**Usage**

```r
ma_d_order2(k, N = NULL, d = NULL, delta = NULL, var_d = NULL, var_d_c = NULL, ma_type = c("bb", "ic", "ad"), sample_id = NULL, citekey = NULL, moderators = NULL, moderator_type = "simple", construct_x = NULL, construct_y = NULL, construct_order = NULL, data = NULL, control = control_psychmeta(), ...)
```

**Arguments**

- **k**: Vector or column name of meta-analyses’ k values.
- **N**: Vector or column name of meta-analyses’ total sample sizes (optional).
- **d**: Vector or column name of mean observed $d$ values.
delta Vector or column name of mean corrected $d$ values.
var_d Vector or column name of observed variances of observed $d$ values.
var_d_c Vector or column name of observed variances of corrected $d$ values.
ma_type Type of meta-analyses being analyzed: "bb" (barebones), "ic" (individual correction), or "ad" (artifact distribution).
sample_id Vector or column name of study ID labels.
citekey Optional vector of bibliographic citation keys for samples/studies in the meta-analysis (if multiple citekeys pertain to a given effect size, combine them into a single string entry with comma delimiters (e.g., "citekey1,citekey2").
moderators Matrix or column names of moderator variables to be used in the meta-analysis (can be a vector in the case of one moderator).
moderator_type Type of moderator analysis ("none", "simple", or "hierarchical").
construct_x Vector or column name of construct names for X.
construct_y Vector or column name of construct names for Y.
construct_order Vector indicating the order in which variables should be arranged, with variables listed earlier in the vector being preferred for designation as X.
data Data frame containing columns whose names may be provided as arguments to vector arguments and/or moderators.
control Output from the control_psychmeta() function or a list of arguments controlled by the control_psychmeta() function. Ellipsis arguments will be screened for internal inclusion in control.

Value
A nested tabular object of the class "ma_psychmeta".

Description
This function computes bare-bones meta-analyses of any effect size using user-supplied effect error variances.

Usage
ma_generic(
es, 
n, 
var_e, 
sample_id = NULL,
Arguments

es
- Vector or column name of observed effect sizes.

n
- Vector or column name of sample sizes.

var_e
- Vector or column name of error variances.

sample_id
- Optional vector of identification labels for samples/studies in the meta-analysis.

citekey
- Optional vector of bibliographic citation keys for samples/studies in the meta-analysis (if multiple citekeys pertain to a given effect size, combine them into a single string entry with comma delimiters (e.g., "citekey1,citekey2"). When TRUE, program will use sample-size weights, error variances estimated from the mean effect size, maximum likelihood variances, and normal-distribution confidence and credibility intervals.

construct_x, construct_y
- Vector of construct names for constructs designated as "X" and as "Y".

group1, group2
- Vector of groups' names associated with effect sizes that represent pairwise contrasts.

wt_type
- Type of weight to use in the meta-analysis: native options are "sample_size" and "inv_var" (inverse error variance). Supported options borrowed from metafor are "DL", "HE", "HS", "SJ", "ML", "REML", "EB", and "PM" (see metafor documentation for details about the metafor methods).

moderators
- Matrix of moderator variables to be used in the meta-analysis (can be a vector in the case of one moderator).

cat_moderators
- Logical scalar or vector identifying whether variables in the moderators argument are categorical variables (TRUE) or continuous variables (FALSE).

moderator_type
- Type of moderator analysis ("none", "simple", or "hierarchical").

data
- Data frame containing columns whose names may be provided as arguments to vector arguments and/or moderators.

control
- Output from the control_psychmeta() function or a list of arguments controlled by the control_psychmeta() function. Ellipsis arguments will be screened for internal inclusion in control.
weights

Optional vector of weights to be used. When weights is non-NULL, these weights override the argument supplied to wt_type.

Further arguments to be passed to functions called within the meta-analysis.

Value

A nested tabular object of the class "ma_psychmeta".

Examples

es <- c(.3, .5, .8)
n <- c(100, 200, 150)
var_e <- 1 / n
ma_obj <- ma_generic(es = es, n = n, var_e = var_e)
ma_obj
summary(ma_obj)

ma_r

Meta-analysis of correlations

Description

The ma_r_bb, ma_r_ic, and ma_r_ad functions implement bare-bones, individual-correction, and artifact-distribution correction methods for correlations, respectively. The ma_r function is the master function for meta-analyses of correlations - it facilitates the computation of bare-bones, artifact-distribution, and individual-correction meta-analyses of correlations for any number of construct pairs. When artifact-distribution meta-analyses are performed, ma_r will automatically extract the artifact information from a database and organize it into the requested type of artifact distribution object (i.e., either Taylor series or interactive artifact distributions). ma_r is also equipped with the capability to clean databases containing inconsistently recorded artifact data, impute missing artifacts (when individual-correction meta-analyses are requested), and remove dependency among samples by forming composites or averaging effect sizes and artifacts. The automatic compositing features in ma_r are employed when sample_ids and/or construct names are provided.

Usage

ma_r(
  rxyi,
  n,
  n_adj = NULL,
  sample_id = NULL,
  citekey = NULL,
  ma_method = c("bb", "ic", "ad"),
  ad_type = c("tsa", "int"),
  correction_method = "auto",
  construct_x = NULL,
  construct_y = NULL,
  facet_x = NULL,
ma_r

facet_y = NULL,
measure_x = NULL,
measure_y = NULL,
construct_order = NULL,
wt_type = c("sample_size", "inv_var_mean", "inv_var_sample", "DL", "HE", "HS", "SJ",
               "ML", "REML", "EB", "PM"),
correct_bias = TRUE,
correct_rel = NULL,
correct_rxx = TRUE,
correct_ryy = TRUE,
correct_rr = NULL,
correct_rr_x = TRUE,
correct_rr_y = TRUE,
indirect_rr = NULL,
indirect_rr_x = TRUE,
indirect_rr_y = TRUE,
rxx = NULL,
rxx_restricted = TRUE,
rxx_type = "alpha",
k_items_x = NULL,
ryy = NULL,
ryy_restricted = TRUE,
ryy_type = "alpha",
k_items_y = NULL,
ux = NULL,
ux_observed = TRUE,
uy = NULL,
uy_observed = TRUE,
sign_rz = NULL,
sign_rxz = 1,
sign_ryz = 1,
moderators = NULL,
cat_moderators = TRUE,
moderator_type = c("simple", "hierarchical", "none"),
supplemental_ads = NULL,
data = NULL,
control = control_psychmeta(),
"

ma_r_ad(
    ma_obj,
    ad_obj_x = NULL,
    ad_obj_y = NULL,
correction_method = "auto",
use_ic_ads = c("tsa", "int"),
correct_rxx = TRUE,
correct_ryy = TRUE,
`ma_r_bb`

```r
ma_r_bb(
  r,
  n,
  n_adj = NULL,
  sample_id = NULL,
  citekey = NULL,
  wt_type = c("sample_size", "inv_var_mean", "inv_var_sample", "DL", "HE", "HS", "SJ",
              "ML", "REML", "EB", "PM"),
  correct_bias = TRUE,
  moderators = NULL,
  cat_moderators = TRUE,
  moderator_type = c("simple", "hierarchical", "none"),
  data = NULL,
  control = control_psychmeta(),
  ...
)
```

`ma_r_ic`

```r
ma_r_ic(
  rxyi,
  n,
  n_adj = NULL,
  sample_id = NULL,
  citekey = NULL,
  wt_type = c("sample_size", "inv_var_mean", "inv_var_sample", "DL", "HE", "HS", "SJ",
              "ML", "REML", "EB", "PM"),
  correct_bias = TRUE,
  correct_rxx = TRUE,
  correct_ryy = TRUE,
  correct_rr_x = TRUE,
  correct_rr_y = TRUE,
  indirect_rr_x = TRUE,
  indirect_rr_y = TRUE,
  rxx = NULL,
  rxx_restricted = TRUE,
  rxx_type = "alpha",
  k_items_x = NULL,
  ryy = NULL,
  ryy_restricted = TRUE,
  ...
)```
ma_r

ryy_type = "alpha",
k_items_y = NULL,
ux = NULL,
ux_observed = TRUE,
uy = NULL,
uy_observed = TRUE,
sign_rxz = 1,
sign_ryz = 1,
moderators = NULL,
cat_moderators = TRUE,
moderator_type = c("simple", "hierarchical", "none"),
supplemental_ads_x = NULL,
supplemental_ads_y = NULL,
data = NULL,
control = control_psychmeta(),
...)

Arguments

rxyi, r Vector or column name of observed correlations. The r argument is used with the \texttt{ma_r_bb} function and the \texttt{rxyi} argument is used with \texttt{ma_r} and \texttt{ma_r_ic} function (i.e., the function in which corrections are applied). \textit{NOTE:} Beginning in \texttt{psychmeta} version 2.5.2, \texttt{rxyi} values of exactly 0 in individual-correction meta-analyses are replaced with a functionally equivalent value via the \texttt{zero_substitute} argument for \texttt{control_psychmeta} to facilitate the estimation of corrected error variances.

n Vector or column name of sample sizes.
n_adj Optional: Vector or column name of sample sizes adjusted for sporadic artifact corrections.
sample_id Optional vector of identification labels for samples/studies in the meta-analysis.
citekey Optional vector of bibliographic citation keys for samples/studies in the meta-analysis (if multiple citekeys pertain to a given effect size, combine them into a single string entry with comma delimiters (e.g., "citekey1,citekey2").
ma_method Method to be used to compute the meta-analysis: "bb" (barebones), "ic" (individual correction), or "ad" (artifact distribution).
ad_type For when \texttt{ma_method} is "ad". Specifies the type of artifact distribution to use: "int" or "tsa".
correction_method For when \texttt{ma_method} is "ad". Character scalar or a square matrix with the collective levels of \texttt{construct_x} and \texttt{construct_y} as row names and column names. Select one of the following methods for correcting artifacts: "auto", "meas", "uvdr", "uvir", "bvdrr", "bvirr", "rbOrig", "rb1Orig", "rb2Orig", "rbAdj", "rb1Adj", and "rb2Adj" (note: "rb1Orig", "rb2Orig", "rb1Adj", and "rb2Adj" can only be used when Taylor series artifact distributions are provided and "rbOrig" and "rbAdj" can only be used when iterative artifact distributions are provided). See "Details" of \texttt{ma_r_ad} for descriptions of the available methods.
construct_x, construct_y
Vector of construct names for constructs initially designated as "X" or as "Y".

facet_x, facet_y
Vector of facet names for constructs initially designated as "X" or as "Y". Facet names "global", "overall", and "total" are reserved to indicate observations that represent effect sizes that have already been composited or that represent construct-level measurements rather than facet-level measurements. To avoid double-compositing, any observation with one of these reserved names will only be eligible for auto-compositing with other such observations and will not be combined with narrow facets.

measure_x, measure_y
Vector of names for measures associated with constructs initially designated as "X" or as "Y".

construct_order
Vector indicating the order in which variables should be arranged, with variables listed earlier in the vector being preferred for designation as X.

wt_type
Type of weight to use in the meta-analysis: options are "sample_size", "inv_var_mean" (inverse variance computed using mean effect size), and "inv_var_sample" (inverse variance computed using sample-specific effect sizes). Supported options borrowed from metafor are "DL", "HE", "HS", "SJ", "ML", "REML", "EB", and "PM" (see metafor documentation for details about the metafor methods).

correct_bias
Logical scalar that determines whether to correct correlations for small-sample bias (TRUE) or not (FALSE).

correct_rel
Optional named vector that supersedes correct_rxx and correct_ryy. Names should correspond to construct names in construct_x and construct_y to determine which constructs should be corrected for unreliability.

correct_rxx, correct_ryy
Logical scalar or vector that determines whether to correct the X or Y variable for measurement error (TRUE) or not (FALSE).

correct_rr
Optional named vector that supersedes correct_rr_x and correct_rr_y. Names should correspond to construct names in construct_x and construct_y to determine which constructs should be corrected for range restriction.

correct_rr_x
Logical scalar, logical vector, or column name determining whether each correlation in rxyi should be corrected for range restriction in X (TRUE) or not (FALSE). If using artifact distribution methods, this must be a scalar value.

correct_rr_y
Logical scalar, logical vector, or column name determining whether each correlation in rxyi should be corrected for range restriction in Y (TRUE) or not (FALSE). If using artifact distribution methods, this must be a scalar value.

indirect_rr
Optional named vector that supersedes indirect_rr_x and indirect_rr_y. Names should correspond to construct names in construct_x and construct_y to determine which constructs should be corrected for indirect range restriction.

indirect_rr_x
Logical vector or column name determining whether each correlation in rxyi should be corrected for indirect range restriction in X (TRUE) or not (FALSE). Superseded in evaluation by correct_rr_x (i.e., if correct_rr_x == FALSE, the value supplied for indirect_rr_x is disregarded).
indirect_rr_y  Logical vector or column name determining whether each correlation in \( r_{xyi} \) should be corrected for indirect range restriction in Y (TRUE) or not (FALSE). Superseded in evaluation by correct_rr_y (i.e., if correct_rr_y == FALSE, the value supplied for indirect_rr_y is disregarded).

rxx  Vector or column name of reliability estimates for X.

rxx_restricted  Logical vector or column name determining whether each element of rxx is an incumbent reliability (TRUE) or an applicant reliability (FALSE).

rxx_type, ryy_type  String vector identifying the types of reliability estimates supplied. Acceptable reliability types are:

- internal_consistency  A generic designation for internal-consistency reliability estimates derived from responses to a single test administration.
- multiple_administrations  A generic designation for reliability estimates derived from multiple administrations of a test.
- alpha  Coefficient alpha.
- lambda  Generic designation for a Guttman’s lambda coefficient.
- lambda1  Guttman’s lambda 1 coefficient.
- lambda2  Guttman’s lambda 2 coefficient.
- lambda3  Guttman’s lambda 3 coefficient.
- lambda4  Guttman’s lambda 4 coefficient.
- lambda5  Guttman’s lambda 5 coefficient.
- lambda6  Guttman’s lambda 6 coefficient.
- omega  Omega coefficient indicating the proportion variance in a variable accounted for by modeled latent factors.
- icc  Intraclass correlation coefficient.
- interrater_r  Inter-rater correlation coefficient.
- interrater_r_sb  Inter-rater correlation coefficient, stepped up with the Spearman-Brown formula.
- splithalf  Split-half reliability coefficient.
• split-half_{sb}
  Split-half reliability coefficient, corrected toward the full test length with
  the Spearman-Brown formula.
• retest
  Test-retest reliability coefficient.
• parallel
  Parallel-forms reliability coefficient with tests taken during the same testing
  session.
• alternate
  Alternate-forms reliability coefficient with tests taken during the same testing
  session.
• parallel_{delayed}
  Parallel-forms reliability coefficient with tests taken during separate testing
  sessions with a time delay in between.
• alternate_{delayed}
  Alternate-forms reliability coefficient with tests taken during separate testing
  sessions with a time delay in between.

k_items_x, k_items_y
  Numeric vector identifying the number of items in each scale.

rry
  Vector or column name of reliability estimates for Y.

rry_restricted
  Logical vector or column name determining whether each element of coderyy is
  an incumbent reliability (TRUE) or an applicant reliability (FALSE).

ux
  Vector or column name of u ratios for X.

ux_observed
  Logical vector or column name determining whether each element of ux is an
  observed-score u ratio (TRUE) or a true-score u ratio (FALSE).

uy
  Vector or column name of u ratios for Y.

uy_observed
  Logical vector or column name determining whether each element of uy is an
  observed-score u ratio (TRUE) or a true-score u ratio (FALSE).

sign_rz
  Optional named vector that supersedes sign_rxz and sign_ryz. Names should
  correspond to construct names in construct_x and construct_y to determine
  the sign of each construct’s relationship with the selection mechanism.

sign_rxz
  Sign of the relationship between X and the selection mechanism (for use with
  bvirr corrections only).

sign_ryz
  Sign of the relationship between Y and the selection mechanism (for use with
  bvirr corrections only).

moderators
  Either (1) a vector of column names in data of moderator variables to be used
  in the meta-analysis (names can be quoted or unquoted), or (2) a vector, data
  frame, or matrix containing moderator variables.

cat_moderators
  Either (1) A character vector listing the variable names in moderators that are
  categorical, or (2) a logical scalar or vector identifying whether each variable in
  moderators is categorical (TRUE) or continuous (FALSE).

moderator_type
  Type of moderator analysis: "none" means that no moderators are to be used,
  "simple" means that moderators are to be examined one at a time, and "hierarchi-
  cal" means that all possible combinations and subsets of moderators are to be
  examined.
**supplemental_ads**
For *ma_r* only: Named list (named according to the constructs included in the meta-analysis) of supplemental artifact distribution information from studies not included in the meta-analysis. This is a list of lists, where the elements of a list associated with a construct are named like the arguments of the `create_ad()` function.

**data**
Data frame containing columns whose names may be provided as arguments to vector arguments and/or moderators.

**control**
Output from the `control_psychmeta()` function or a list of arguments controlled by the `control_psychmeta()` function. Ellipsis arguments will be screened for internal inclusion in `control`.

... Further arguments to be passed to functions called within the meta-analysis.

**ma_obj**
For *ma_r_ad* only: Meta-analysis object of correlations or *d* values (regardless of input metric, output metric will be *r*).

**ad_obj_x**
For *ma_r_ad* only: Artifact-distribution object for the X variable (output of the `create_ad` function). If *ma_obj* is of the class *ma_master* (i.e., the output of *ma_r* or *ma_d*), the object supplied for *ad_obj_x* must be a named list of artifact distributions with names corresponding to the "X" constructs in the meta-analyses contained within *ma_obj*.

**ad_obj_y**
For *ma_r_ad* only: Artifact-distribution object for the Y variable (output of the `create_ad` function). If *ma_obj* is of the class *ma_master*, the object supplied for *ad_obj_y* must be a named list of artifact distributions with names corresponding to the "Y" constructs in the meta-analyses contained within *ma_obj*.

**use_ic_ads**
For *ma_r_ad* only: Determines whether artifact distributions should be extracted from the individual correction results in *ma_obj*. Only evaluated when *ad_obj_x* or *ad_obj_y* is NULL and *ma_obj* does not contain individual correction results. Use one of the following commands: `tsa` to use the Taylor series method or `int` to use the interactive method.

**supplemental_ads_x**, **supplemental_ads_y**
For *ma_r_ic* only: List supplemental artifact distribution information from studies not included in the meta-analysis. The elements of this list are named like the arguments of the `create_ad()` function.

**Details**

The options for `correction_method` are:

- "auto"
  Automatic selection of the most appropriate correction procedure, based on the available artifacts and the logical arguments provided to the function. (default)

- "meas"
  Correction for measurement error only.

- "uvdrr"
  Correction for univariate direct range restriction (i.e., Case II). The choice of which variable to correct for range restriction is made using the `correct_rr_x` and `correct_rr_y` arguments.
• "uvirr"
  Correction for univariate indirect range restriction (i.e., Case IV). The choice of which variable to correct for range restriction is made using the `correct_rr_x` and `correct_rr_y` arguments.

• "bvdrr"
  Correction for bivariate direct range restriction. Use with caution: This correction is an approximation only and is known to have a positive bias.

• "bvirr"
  Correction for bivariate indirect range restriction (i.e., Case V).

• "rbOrig"
  Not recommended: Raju and Burke’s version of the correction for direct range restriction, applied interactively. We recommend using "uvdrr" instead.

• "rbAdj"
  Not recommended: Raju and Burke’s version of the correction for direct range restriction, applied interactively. Adjusted to account for range restriction in the reliability of the Y variable. We recommend using "uvdrr" instead.

• "rb1Orig"
  Not recommended: Raju and Burke’s version of the correction for direct range restriction, applied using their TSA1 method. We recommend using "uvdrr" instead.

• "rb1Adj"
  Not recommended: Raju and Burke’s version of the correction for direct range restriction, applied using their TSA1 method. Adjusted to account for range restriction in the reliability of the Y variable. We recommend using "uvdrr" instead.

• "rb2Orig"
  Not recommended: Raju and Burke’s version of the correction for direct range restriction, applied using their TSA2 method. We recommend using "uvdrr" instead.

• "rb2Adj"
  Not recommended: Raju and Burke’s version of the correction for direct range restriction, applied using their TSA2 method. Adjusted to account for range restriction in the reliability of the Y variable. We recommend using "uvdrr" instead.

Value

A nested tabular object of the class "ma_psychmeta". Components of output tables for bare-bones meta-analyses:

• `pair_id`
  Unique identification number for each construct pairing.

• `construct_x`
  Name of the variable analyzed as construct X.

• `construct_y`
  Name of the variable analyzed as construct Y.

• `analysis_id`
  Unique identification number for each analysis.

• `analysis_type`
  Type of moderator analyses: Overall, Simple Moderator, or Hierarchical Moderator.
• **k**
  Number of effect sizes meta-analyzed.

• **N**
  Total sample size of all effect sizes in the meta-analysis.

• **mean_r**
  Mean observed correlation.

• **var_r**
  Weighted variance of observed correlations.

• **var_e**
  Predicted sampling-error variance of observed correlations.

• **var_res**
  Variance of observed correlations after removing predicted sampling-error variance.

• **sd_r**
  Square root of var_r.

• **se_r**
  Standard error of mean_r.

• **sd_e**
  Square root of var_e.

• **sd_res**
  Square root of var_res.

• **CI_LL_XX**
  Lower limit of the confidence interval around mean_r, where "XX" represents the confidence level as a percentage.

• **CI_UL_XX**
  Upper limit of the confidence interval around mean_r, where "XX" represents the confidence level as a percentage.

• **CR_LL_XX**
  Lower limit of the credibility interval around mean_r, where "XX" represents the credibility level as a percentage.

• **CR_UL_XX**
  Upper limit of the credibility interval around mean_r, where "XX" represents the credibility level as a percentage.

Components of output tables for individual-correction meta-analyses:

• **pair_id**
  Unique identification number for each construct pairing.

• **construct_x**
  Name of the variable analyzed as construct X.

• **construct_y**
  Name of the variable analyzed as construct Y.

• **analysis_id**
  Unique identification number for each analysis.
• analysis_type
  Type of moderator analyses: Overall, Simple Moderator, or Hierarchical Moderator.

• \( k \)
  Number of effect sizes meta-analyzed.

• \( N \)
  Total sample size of all effect sizes in the meta-analysis.

• mean_r
  Mean observed correlation.

• var_r
  Weighted variance of observed correlations.

• var_e
  Predicted sampling-error variance of observed correlations.

• var_res
  Variance of observed correlations after removing predicted sampling-error variance.

• sd_r
  Square root of var_r.

• se_r
  Standard error of mean_r.

• sd_e
  Square root of var_e.

• sd_res
  Square root of var_res.

• mean_rho
  Mean artifact-corrected correlation.

• var_r_c
  Variance of artifact-corrected correlations.

• var_e_c
  Predicted sampling-error variance of artifact-corrected correlations.

• var_rho
  Variance of artifact-corrected correlations after removing predicted sampling-error variance.

• sd_r_c
  Square root of var_r_c.

• se_r_c
  Standard error of mean_rho.

• sd_e_c
  Square root of var_e_c.

• sd_rho
  Square root of var_rho.

• CI_LL_XX
  Lower limit of the confidence interval around mean_rho, where "XX" represents the confidence level as a percentage.
Components of output tables for artifact-distribution meta-analyses:

- **pair_id**
  Unique identification number for each construct pairing.

- **construct_x**
  Name of the variable analyzed as construct X.

- **construct_y**
  Name of the variable analyzed as construct Y.

- **analysis_id**
  Unique identification number for each analysis.

- **analysis_type**
  Type of moderator analyses: Overall, Simple Moderator, or Hierarchical Moderator.

- **k**
  Number of effect sizes meta-analyzed.

- **N**
  Total sample size of all effect sizes in the meta-analysis.

- **mean_r**
  Mean observed correlation.

- **var_r**
  Weighted variance of observed correlations.

- **var_e**
  Predicted sampling-error variance of observed correlations.

- **var_art**
  Amount of variance in observed correlations that is attributable to measurement-error and range-restriction artifacts.

- **var_pre**
  Total predicted artifactual variance (i.e., the sum of var_e and var_art).

- **var_res**
  Variance of observed correlations after removing predicted sampling-error variance and predicted artifact variance.

- **sd_r**
  Square root of var_r.

- **se_r**
  Standard error of mean_r.
• sd_e
  Square root of var_e.
• sd_art
  Square root of var_art.
• sd_pre
  Square root of var_pre.
• sd_res
  Square root of var_res.
• mean_rho
  Mean artifact-corrected correlation.
• var_r_c
  Weighted variance of observed correlations corrected to the metric of rho.
• var_e_c
  Predicted sampling-error variance of observed correlations corrected to the metric of rho.
• var_art_c
  Amount of variance in observed correlations that is attributable to measurement-error and range-restriction artifacts corrected to the metric of rho.
• var_pre_c
  Total predicted artifactual variance (i.e., the sum of var_e and var_art) corrected to the metric of rho.
• var_rho
  Variance of artifact-corrected correlations after removing predicted sampling-error variance and predicted artifact variance.
• sd_r_c
  Square root of var_r corrected to the metric of rho.
• se_r_c
  Standard error of mean_r corrected to the metric of rho.
• sd_e_c
  Square root of var_e corrected to the metric of rho.
• sd_art_c
  Square root of var_art corrected to the metric of rho.
• sd_pre_c
  Square root of var_pre corrected to the metric of rho.
• sd_rho
  Square root of var_rho.
• CI_LL_XX
  Lower limit of the confidence interval around mean_rho, where "XX" represents the confidence level as a percentage.
• CI_UL_XX
  Upper limit of the confidence interval around mean_rho, where "XX" represents the confidence level as a percentage.
• CR_LL_XX
  Lower limit of the credibility interval around mean_rho, where "XX" represents the credibility level as a percentage.
• CR_UL_XX
  Upper limit of the credibility interval around mean_rho, where "XX" represents the credibility level as a percentage.

Note

The difference between "rb" methods with the "orig" and "adj" suffixes is that the original does not account for the impact of range restriction on criterion reliabilities, whereas the adjusted procedure attempts to estimate the applicant reliability information for the criterion. The "rb" procedures are included for posterity: We strongly recommend using the "uvdrr" procedure to appropriately correct for univariate range restriction.

References


Examples

```r
## Not run:
## The 'ma_r' function can compute multi-construct bare-bones meta-analyses:
ma_obj <- ma_r(rxyi = rxyi, n = n, rxx = rxxi, ryy = ryyi,
               construct_x = x_name, construct_y = y_name, sample_id = sample_id,
               moderators = moderator, data = data_r_meas_multi)
summary(ma_obj)

## It can also perform multiple individual-correction meta-analyses:
ma_obj <- ma_r(ma_method = "ic", rxyi = rxyi, n = n, rxx = rxxi, ryy = ryyi,
               construct_x = x_name, construct_y = y_name, sample_id = sample_id,
               moderators = moderator, data = data_r_meas_multi)
summary(ma_obj)

ma_obj$meta_tables[[1]]$individual_correction$true_score

## And 'ma_r' can also curate artifact distributions and compute multiple
## artifact-distribution meta-analyses:
ma_obj <- ma_r(ma_method = "ad", ad_type = "int", rxyi = rxyi, n = n, rxx = rxxi, ryy = ryyi,
               correct_rr_x = FALSE, correct_rr_y = FALSE,
               construct_x = x_name, construct_y = y_name, sample_id = sample_id,
               clean_artifacts = FALSE, impute_artifacts = FALSE,
               moderators = moderator, data = data_r_meas_multi)
summary(ma_obj)
ma_obj$meta_tables[[1]]$artifact_distribution$true_score
```
## Even if no studies in the database provide artifact information, ## pre-specified artifact distributions from previous meta-analyses can still be used! (These results should match the previous example.)

```r
ma_obj <- ma_r(ma_method = "ad", rxyi = rxyi, n = n,
               correct_rr_x = FALSE, correct_rr_y = FALSE,
               construct_x = x_name, construct_y = y_name, sample_id = sample_id,
               clean_artifacts = FALSE, impute_artifacts = FALSE,
               moderators = moderator, data = data_r measles multi,
               supplemental_ads =
               list(X = list(mean_qxi = 0.8927818, var_qxi = 0.0008955520, k_qxi = 40,
                       mean_n_qxi = 11927 / 40, qxi_dist_type = "alpha"),
                    Y = list(mean_qxi = 0.8941266, var_qxi = 0.0009367234, k_qxi = 40,
                       mean_n_qxi = 11927 / 40, qxi_dist_type = "alpha"),
                    Z = list(mean_qxi = 0.8962108, var_qxi = 0.0007840593, k_qxi = 40,
                       mean_n_qxi = 11927 / 40, qxi_dist_type = "alpha")))
summary(ma_obj)
ma_obj$meta_tables[[1]]$artifact_distribution$true_score
```

## Artifact information may also be supplied by passing "ad_obj" class objects with the "supplemental_ads" argument.

## Create a list of artifact-distribution objects:
```r
ad_list <- create_ad_list(n = n, rxx = rxxi, ryy = ryyi,
                           construct_x = x_name, construct_y = y_name,
                           sample_id = sample_id,
                           data = data_r measles multi)
ad_list <- setNames(ad_list$ad_x, ad_list$construct_x)
```

## Run the artifact-distribution meta-analysis:
```r
ma_obj <- ma_r(ma_method = "ad", rxyi = rxyi, n = n,
               correct_rr_x = FALSE, correct_rr_y = FALSE,
               construct_x = x_name, construct_y = y_name, sample_id = sample_id,
               clean_artifacts = FALSE, impute_artifacts = FALSE,
               moderators = moderator, data = data_r measles multi,
               supplemental_ads = ad_list)
summary(ma_obj)
ma_obj$meta_tables[[1]]$artifact_distribution$true_score
```

## Artifact information from studies not included in the meta-analysis can also be used to make corrections. Passing artifact information with the 'supplemental_ads' argument allows for additional artifact values and/or means and variances of artifacts to be used.

## The 'supplemental_ads' analysis below gives the same results as the prior meta-analysis.

```r
x_ids <- c(data_r measles multi$x_name, data_r measles multi$y_name) == "X"
rxxi <- c(data_r measles multi$rxxi, data_r measles multi$ryyi)[x_ids]
n_rxxi = c(data_r measles multi$n, data_r measles multi$n)[x_ids]

y_ids <- c(data_r measles multi$x_name, data_r measles multi$y_name) == "Y"
ryyi <- c(data_r measles multi$rxxi, data_r measles multi$ryyi)[y_ids]
n_ryyi = c(data_r measles multi$n, data_r measles multi$n)[y_ids]

z_ids <- c(data_r measles multi$x_name, data_r measles multi$y_name) == "Z"
rzzi <- c(data_r measles multi$rxxi, data_r measles multi$ryyi)[z_ids]
```
n_rzzi = c(data_r_meas_multi$n, data_r_meas_multi$n)[z_ids]

ma_obj <- ma_r(ma_method = "ad", rxyi = rxyi, n = n,
               correct_rr_x = FALSE, correct_rr_y = FALSE,
               construct_x = x_name, construct_y = y_name,
               moderators = moderator, sample_id = sample_id, data = data_r_meas_multi,
               supplemental_ads = list(X = list(rxxi = rxxi, n_rxxi = n_rxxi, wt_rxxi = n_rxxi),
                                      Y = list(ryyi = ryyi, n_ryyi = n_ryyi, wt_ryyi = n_ryyi),
                                      Z = list(rzzi = rzzi, n_rzzi = n_rzzi, wt_rzzi = n_rzzi)))

summary(ma_obj)

If 'use_all_arts' is set to TRUE, artifacts from studies without valid correlations
will be used to inform artifact distributions. Below, correlations and artifacts
are provided by non-overlapping sets of studies.

dat1 <- dat2 <- data_r_meas_multi
dat1$rxxi <- dat1$ryyi <- NA
dat2$rxyi <- NA
dat2$sample_id <- dat2$sample_id + 40
dat <- rbind(dat1, dat2)

ma_obj <- ma_r(ma_method = "ad", rxyi = rxyi, n = n, rxxi = rxxi, ryyi = ryyi,
               correct_rr_x = FALSE, correct_rr_y = FALSE,
               construct_x = x_name, construct_y = y_name,
               sample_id = sample_id, moderators = moderator,
               use_all_arts = TRUE, data = dat)

summary(ma_obj)

### Demonstration of ma_r_bb ###
## Example analysis using data from Gonzalez-Mule et al. (2014):
## Not correcting for bias and using normal distributions to compute uncertainty intervals
## allows for exact replication of the results reported in the text:
ma_r_bb(r = rxyi, n = n, correct_bias = FALSE, conf_method = "norm", cred_method = "norm",
         data = data_r_gonzalezmule_2014)

## Using hs_override = TRUE allows one to easily implement the traditional Hunter-Schmidt method:
ma_r_bb(r = rxyi, n = n, hs_override = TRUE, data = data_r_gonzalezmule_2014)

## With hs_override = FALSE, the program defaults will compute unbiased variances and use
## t-distributions to estimate confidence and credibility intervals - these settings make
## a noticeable difference for small studies like the textbook example:
ma_r_bb(r = rxyi, n = n, hs_override = FALSE, data = data_r_gonzalezmule_2014)

### Demonstration of ma_r_ic ###
## Simulated example satisfying the assumptions of the Case IV
## range-restriction correction (parameter values: mean_rho = .3, sd_rho = .15):
ma_r_ic(rxyi = rxyi, n = n, rxxi = rxxi, ryyi = ryyi, ux = ux, data = data_r_uvirr)
## Simulated example satisfying the assumptions of the Case V
## range-restriction correction
ma_r_ic(rxyi = rxyi, n = n, rxx = rxxi, ryy = ryyi,
    rxx_type = "parallel", ryy_type = "parallel",
    ux = ux, uy = uy, data = data_r_bvirr)

## Published example from Gonzalez-Mule et al. (2014)
ma_r_ic(rxyi = rxyi, n = n, hs_override = TRUE, data = data_r_gonzalezmule_2014,
    rxx = rxxi, ryy = ryyi, ux = ux, indirect_rr_x = TRUE,
    moderators = c("Rating source", "Published", "Type", "Complexity"))

### Demonstration of ma_r_ad ###
## Compute barebones meta-analysis
ma_obj <- ma_r_bb(r = rxyi, n = n, correct_bias = FALSE,
    conf_method = "norm", cred_method = "norm", data = data_r_mcdaniel_1994)

## Construct artifact distribution for X
ad_obj_x <- create_ad(ad_type = "tsa", mean_rxxi = data_r_mcdaniel_1994$Mrxxi[1],
    var_rxxi = data_r_mcdaniel_1994$SDrxxi[1]^2, .5,
    ux = data_r_mcdaniel_1994$ux,
    wt_ux = data_r_mcdaniel_1994$'ux frequency')

## Construct artifact distribution for Y
ad_obj_y <- create_ad(ad_type = "tsa", rxxi = data_r_mcdaniel_1994$ryyi,
    wt_rxxi = data_r_mcdaniel_1994$'ryyi frequency')

## Compute artifact-distribution meta-analysis, correcting for measurement error only
ma_r_ad(ma_obj = ma_obj, ad_obj_x = ad_obj_x, ad_obj_y = ad_obj_y, correction_method = "meas")

## Compute artifact-distribution meta-analysis, correcting for univariate direct range restriction
ma_r_ad(ma_obj = ma_obj, ad_obj_x = ad_obj_x, ad_obj_y = ad_obj_y, correction_method = "uvdrr",
    correct_rr_y = FALSE, indirect_rr_x = FALSE)

# The results of ma_r() can also be corrected using artifact distributions
ma_obj <- ma_r(ma_method = "bb", rxyi = rxyi, n = n,
    construct_x = x_name, construct_y = y_name, sample_id = sample_id,
    moderators = moderator, data = data_r_meas_multi)

# The create_ad_list function can be used to generate batches of artifact-distribution objects.
# Here is an example in which one distribution is created per construct.
ad_tibble <- create_ad_list(n = n, rxx = rxxi, ryy = ryyi,
    construct_x = x_name, construct_y = y_name, sample_id = sample_id,
    moderators = moderator, data = data_r_meas_multi)

# Passing that collection of distributions to ma_r_ad() corrects 'ma_obj' for artifacts:
ma_obj_tibble <- ma_r_ad(ma_obj = ma_obj, ad_obj_x = ad_tibble, ad_obj_y = ad_tibble)
summary(ma_obj_tibble)
ma_obj_tibble$meta_tables[[1]]$artifact_distribution$true_score
# The same outcomes as the previous example can be achieved by passing a named list of artifact information, with each element bearing the name of a construct:
```
ad_list <- setNames(ad_tibble$ad_x, ad_tibble$construct_x)
ma_obj_list <- ma_r_ad(ma_obj = ma_obj,
ad_obj_x = ad_list, ad_obj_y = ad_list)
summary(ma_obj_list)
ma_obj_list$meta_tables[[1]]$artifact_distribution$true_score
```

# It is also possible to construct artifact distributions in a pairwise fashion.
# For example, if correlations between X and Y and between X and Z are being analyzed, X will get a different distribution for its relationships with Y than with Z. These pairwise distributions are based only on artifact data from specific construct pairs.
```
ad_tibble_pair <- create_ad_list(n = n, rxx = rxxi, ryy = ryyi,
construct_x = x_name, construct_y = y_name,
sample_id = sample_id,
control = control_psychmeta(pairwise_ads = TRUE),
data = data_r_meas_multi)

# Passing these pairwise distributions to ma_r_ad() corrects 'ma_obj' for artifacts:
ma_obj_pair <- ma_r_ad(ma_obj = ma_obj,
ad_obj_x = ad_tibble_pair, ad_obj_y = ad_tibble_pair)
summary(ma_obj_pair)
ma_obj_pair$meta_tables[[1]]$artifact_distribution$true_score
```

# Sometimes moderators have important influence on artifact distributions as well as distributions of effect sizes. When this occurs, moderated artifact distributions can be created to make more appropriate corrections.
```
ad_tibble_mod <- create_ad_list(n = n, rxx = rxxi, ryy = ryyi,
construct_x = x_name, construct_y = y_name,
sample_id = sample_id,
control = control_psychmeta(moderated_ads = TRUE),
moderators = moderator,
data = data_r_meas_multi)

# Passing these moderated distributions to ma_r_ad() corrects 'ma_obj' for artifacts:
ma_obj_mod <- ma_r_ad(ma_obj = ma_obj,
ad_obj_x = ad_tibble_mod, ad_obj_y = ad_tibble_mod)
summary(ma_obj_mod)
ma_obj_mod$meta_tables[[1]]$artifact_distribution$true_score
```

# It is also possible to create pairwise moderated artifact distributions.
```
ad_tibble_pairmod <- create_ad_list(n = n, rxx = rxxi, ryy = ryyi,
construct_x = x_name, construct_y = y_name,
sample_id = sample_id,
control = control_psychmeta(moderated_ads = TRUE,
    pairwise_ads = TRUE),
moderators = moderator,
data = data_r_meas_multi)

# Passing these pairwise moderated distributions to ma_r_ad() corrects 'ma_obj' for artifacts:
ma_obj_pairmod <- ma_r_ad(ma_obj = ma_obj,
ad_obj_x = ad_tibble_pairmod, ad_obj_y = ad_tibble_pairmod)
summary(ma_obj_pairmod)
ma_obj_pairmod$meta_tables[[1]]$artifact_distribution$true_score

# For even more control over which artifact distributions are used in corrections, you can supply
# un-named list of distributions in which the order of distributions corresponds to the order of
# meta-analyses in ma_obj. It is important for the elements to be un-named, as the absence of names
# and the length of the list are the two ways in which ma_r_ad() validates the lists.
ad_list_pairmod_x <- ad_tibble_pairmod$ad_x
ad_list_pairmod_y <- ad_tibble_pairmod$ad_y
# Passing these lists of distributions to ma_r_ad() corrects 'ma_obj' for artifacts:
ma_obj_pairmodlist <- ma_r_ad(ma_obj = ma_obj,
   ad_obj_x = ad_list_pairmod_x, ad_obj_y = ad_list_pairmod_y)
summary(ma_obj_pairmodlist)
ma_obj_pairmodlist$meta_tables[[1]]$artifact_distribution$true_score

## End(Not run)

---

**ma_r_order2**

*Second-order meta-analysis function for correlations*

**Description**

This function computes second-order meta-analysis function for correlations. It supports second-order analyses of bare-bones, artifact-distribution, and individual-correction meta-analyses.

**Usage**

```r
ma_r_order2(
  k,
  N = NULL,
  r = NULL,
  rho = NULL,
  var_r = NULL,
  var_r_c = NULL,
  ma_type = c("bb", "ic", "ad"),
  sample_id = NULL,
  citekey = NULL,
  moderators = NULL,
  moderator_type = "simple",
  construct_x = NULL,
  construct_y = NULL,
  construct_order = NULL,
  data = NULL,
  control = control_psychmeta(),
  ...
)
```
Arguments

k Vector or column name of meta-analyses’ k values.
N Vector or column name of meta-analyses’ total sample sizes (optional).
r Vector or column name of mean observed correlations.
rho Vector or column name of mean corrected correlations.
var_r Vector or column name of observed variances of observed correlations.
var_r_c Vector or column name of observed variances of corrected correlations.
ma_type Type of meta-analyses being analyzed: "bb" (barebones), "ic" (individual correction), or "ad" (artifact distribution).
sample_id Vector or column name of study ID labels.
citekey Optional vector of bibliographic citation keys for samples/studies in the meta-analysis (if multiple citekeys pertain to a given effect size, combine them into a single string entry with comma delimiters (e.g., "citekey1,citekey2").
moderators Matrix or column names of moderator variables to be used in the meta-analysis (can be a vector in the case of one moderator).
moderator_type Type of moderator analysis ("none", "simple", or "hierarchical").
construct_x Vector or column name of construct names for X.
construct_y Vector or column name of construct names for Y.
construct_order Vector indicating the order in which variables should be arranged, with variables listed earlier in the vector being preferred for designation as X.
data Data frame containing columns whose names may be provided as arguments to vector arguments and/or moderators.
control Output from the control_psychmeta() function or a list of arguments controlled by the control_psychmeta() function. Ellipsis arguments will be screened for internal inclusion in control.
... Further arguments to be passed to functions called within the meta-analysis.

Value

A nested tabular object of the class "ma_psychmeta".

Examples

## Analysis of the validity of conscientiousness as a predictor of job performance in East Asia
out <- ma_r_order2(k = k, r = r_bar_i, rho = rho_bar_i, var_r = var_r,
                   var_r_c = NULL, ma_type = c("bb", "ad"),
                   sample_id = NULL, moderators = NULL,
                   construct_x = NULL, construct_y = NULL,
                   data = dplyr::filter(data_r_oh_2009, Predictor == "Conscientiousness"))
summary(out)

## Analysis of the validity of the Big Five traits as predictors of job performance in East Asia
out <- ma_r_order2(k = k, r = r_bar_i, rho = rho_bar_i, var_r = var_r,
merge_simdat_r

merge_simdat_d

Merge multiple "simdat_d_database" class objects

Description

This function allows for multiple simulated databases from simulate_d_database to be merged together into a single database. Merged databases will be assigned moderator variable codes.

Usage

merge_simdat_d(...)

Arguments

... Collection of objects created by the "simulate_d_database" function. Simply enter the database objects as merge_simdat_d(data_obj1, data_obj2, data_obj_3).

Value

A merged database of class simdat_d

merge_simdat_r

Merge multiple "simdat_r_database" class objects

Description

This function allows for multiple simulated databases from simulate_r_database to be merged together into a single database. Merged databases will be assigned moderator variable codes.

Usage

merge_simdat_r(...)
Arguments

... Collection of objects created by the "simulate_r_database" function. Simply enter the database objects as `merge_simdat_r(data_obj1, data_obj2, data_obj_3).

Value

A merged database of class `simdat_r_database`

Description

Write a summary table of meta-analytic results

Usage

```r
metabulate(
  ma_obj,
  file = NULL,
  output_dir = getwd(),
  output_format = c("word", "html", "pdf", "odt", "text", "rmd"),
  show_msd = TRUE,
  show_conf = TRUE,
  show_cred = TRUE,
  show_se = FALSE,
  show_var = FALSE,
  analyses = "all",
  match = c("all", "any"),
  case_sensitive = TRUE,
  ma_method = "ad",
  correction_type = "ts",
  collapse_construct_labels = TRUE,
  bold_headers = TRUE,
  digits = 2L,
  decimal.mark = getOption("OutDec"),
  leading0 = "conditional",
  drop0integer = FALSE,
  neg.sign = "&minus;",
  pos.sign = "figure_html",
  big.mark = "&#8239;",
  big.interval = 3L,
  small.mark = "&#8239;",
  small.interval = 3L,
  na.mark = "&mdash;",
  lgl.mark = c("+", "&minus;"),
  inf.mark = c("+\infin;", "&minus;\infin;"),
)```
conf_format = "brackets",
cred_format = "brackets",
symbol_es = "ES",
caption = "Results of meta-analyses",
header = NULL,
verbose = FALSE,
unicode = NULL,
bib = NULL,
title.bib = NULL,
style = "apa",
additional_citekeys = NULL,
save_build_files = FALSE,
...)

Arguments

ma_obj A psychmeta meta-analysis object.

file The filename (optionally with a subfolder path) for the output file. If NULL, the function will output directly to the R console (also useful if you want to include psychmeta results in a larger RMarkdown document).

output_dir The filepath for the output directory/folder. Defaults to the current working directory.

output_format The format of the output tables. Available options are Word (default), HTML, PDF (requires LaTeX and the unicode-math LaTeX package to be installed), ODT, rmd (Rmarkdown), and text (plain text). You can also specify the full name of another RMarkdown output_format.

show_msd Logical. Should means and standard deviations of effect sizes be shown (default TRUE)

show_conf Logical. Should confidence intervals be shown (default: TRUE)?

show_cred Logical. Should credibility intervals be shown (default: TRUE)?

show_se Logical. Should standard errors be shown (default: FALSE)?

show_var Logical. Should variances be shown (default: FALSE)?

analyses Which analyses to extract references for? See filter_ma for details.

match Match all or any of the filter criteria? See filter_ma for details.

case_sensitive Logical scalar that determines whether character values supplied in analyses should be treated as case sensitive (TRUE, default) or not (FALSE).

ma_method Meta-analytic methods to be included. Valid options are: "ad", "ic", and "bb". Multiple methods are permitted. By default, results are given for one method with order of priority: 1. "ad", 2. "ic", 3. "bb".

correction_type Type of meta-analytic corrections to be included. Valid options are: "ts" (default), "vgx", and "vgy". Multiple options are permitted.
collapse_construct_labels

Should the construct labels for construct pairs with multiple rows of results be simplified so that only the first occurrence of each set of construct names is shown (TRUE; default) or should construct labels be shown for each row of the table (FALSE).

bold_headers

Logical. Should column headers be bolded (default: TRUE)?

digits, decimal.mark, leading0, drop0integer, neg.sign, pos.sign, big.mark, big.interval, small.mark, small.interval, na.mark, lgl.mark, inf.mark

Number formatting arguments. See format_num for details.

conf_format

How should confidence intervals be formatted? Options are:

- parentheses: Bounds are enclosed in parentheses and separated by a comma: (LO, UP).
- brackets: Bounds are enclosed in square brackets and separated by a comma: [LO, UP].
- columns: Bounds are shown in individual columns.

cred_format

How should credibility intervals be formatted? Options are the same as for conf_format above.

symbol_es

For meta-analyses of generic (non-r, non-d) effect sizes, the symbol used for the effect sizes (default: symbol_es = "ES").

caption

Caption to print before tables. Either a character scalar or a named character vector with names corresponding to combinations of ma_method and correction_type (i.e., bb, ic_ts, ad_vgx, etc.).

header

A list of YAML header parameters to pass to render.

verbose

Logical. Should detailed SD and variance components be shown (default: FALSE)?

unicode

Logical. If output_format is "text", should UTF-8 characters be used (defaults to system default).

bib

A BibTeX file containing the citekeys for the meta-analyses. If provided and file is not NULL, a bibliography will be included with the meta-analysis table. See generate_bib for additional arguments controlling the bibliography.

title.bib

The title to give to the bibliography (see bib above). If NULL, defaults to "Sources Contributing to Meta-Analyses".

style

What style should the bibliography (see bib above) be formatted in? Can be a file path or URL for a CSL citation style or the style ID for any style available from the Zotero Style Repository. Defaults to APA style. (Retrieving a style by ID requires an internet connection. If unavailable, references will be rendered in Chicago style.).

additional_citekeys

Additional citekeys to include in the reference list (see bib above).

save_build_files

Should the RMarkdown and BibLaTeX (if any) files used to generate the output be saved (default: FALSE)?

... Additional arguments to pass to render.
**Value**

A list of meta-analysis results _tibbles_ with "caption" and "footnote" attributes.

If file is specified, formatted tables and bibliographies are exported in the requested output_format.

Formatted tables of meta-analytic output.

**See Also**

Other output functions: `generate_bib()`, `metabulate_rmd_helper()`

**Examples**

```r
## Not run:
## Create a results table for meta-analysis of correlations and output to Word:
ma_r_obj <- ma_r(ma_method = "ic", rxyi = rxyi, n = n, rxx = rxxi, ryy = ryyi,
construct_x = x_name, construct_y = y_name,
moderators = moderator, data = data_r_meas_multi)
metabulate(ma_obj = ma_r_obj, file = "meta tables correlations",
output_format = "word", output_dir = tempdir())

## Output to PDF:
metabulate(ma_obj = ma_r_obj, file = "meta tables correlations",
output_format = "pdf", output_dir = tempdir())

## Output to ODT (LibreOffice):
metabulate(ma_obj = ma_r_obj, file = "meta tables correlations",
output_format = "odt", output_dir = tempdir())

## To produce Markdown tables to include inline in an RMarkdown report,
## leave file == NULL and output_format to anything but "text":
ma_table <- metabulate(ma_obj = ma_r_obj, file = NULL, output_format = "rmd")

## Use the metabulate_rmd_helper() function to ensure all symbols render properly.
## Insert the following code as 'as-is' output:
metabulate_rmd_helper()

## Then, add the formatted table to your document using your preferred table
## formatting functions:

### Using just the 'knitr' package, include the following as 'as-is' output:
knitr::kable(ma_table[[1]], caption = attr(ma_table[[1]], "caption"))
cat("\n", attr(ma_table[[1]], "footnote"))

### Using 'knitr' plus the 'kableExtra' package:
knitr::kable(ma_table[[1]], "latex", booktabs = TRUE,
caption = attr(ma_table[[1]], "caption")) \%
\%>
kableExtra::kable_styling(latex_options = c("striped", "hold_position")) \%
\%>
kableExtra::footnote(general = attr(ma_table[[1]], "footnote")

# !!! Note: On Windows, R currently can only handle Unicode characters if kables
# are printed at top-level (e.g., not in an if() statement, in a for() loop,
```
# or in lapply() or map() ). To correctly print Unicode metabulate tables, call
# kable() as a top-level function (as above).

## Create output table for meta-analysis of d values:
```r
ma_d_obj <- ma_d(ma_method = "ic", d = d, n1 = n1, n2 = n2, ryy = ryyi,
                construct_y = construct, data = data_d_meas_multi)
ma_d_obj <- ma_d_ad(ma_obj = ma_d_obj, correct_rr_g = FALSE, correct_rr_y = FALSE)
metabulate(ma_obj = ma_d_obj, file = "meta tables d values", output_dir = tempdir())
```

## Create output table for meta-analysis of generic effect sizes:
```r
dat <- data.frame(es = data_r_meas_multi$rxyi,
                  n = data_r_meas_multi$n,
                  var_e = (1 - data_r_meas_multi$rxyi^2)^2 / (data_r_meas_multi$n - 1))
ma_obj <- ma_generic(es = es, n = n, var_e = var_e, data = dat)
metabulate(ma_obj = ma_obj, file = "meta tables generic es", output_dir = tempdir())
```

## End(Not run)

---

**Description**

The `metabulate` function is used to render meta-analysis results table column headings and footnotes. If `metabulate` is used to render files directly, these are added to the internal RMarkdown document. If you use `metabulate` output in a larger RMarkdown document, use this function to automatically add the necessary lines of code based on your chosen output format.

**Usage**

```r
metabulate_rmd_helper(latex = TRUE, html = TRUE, word_proc = TRUE)
```

**Arguments**

- **latex**
  - Should required commands be included when converting to PDF, LaTeX, and related formats?

- **html**
  - Should required commands be included when converting to HTML and related formats?

- **word_proc**
  - Should required commands be included when converting to Word, ODT, and related formats?

**Value**

Requested commands are printed to the console.
PDF and LaTeX output

If `latex` is `TRUE` and you render to PDF, LaTeX, or other output formats requiring \LaTeX{} (e.g., `beamer_presentation`, see `knitr::is_latex_output`), a YAML metadata block with a `header-includes` argument calling the required `unicode-math` \LaTeX{} package is printed.

An RMarkdown file can only include one `header-includes` metadata entry. If your document already has one, set `latex` to `FALSE` and manually add the `unicode-math` package to your \LaTeX{} header instead.

(Note that `header-includes` is generally discouraged in favor of adding an `include` argument to specific output formats, see `https://bookdown.org/yihui/rmarkdown/pdf-document.html#includes`.)

HTML output

If `html` is `TRUE` and you render to HTML (or related formats, see `knitr::is_html_output`, the following \LaTeX{} math commands are defined:

- `\textit{symit}`
- `\textup{symup}`
- `\textbf{symbfit}`
- `\textbf{symbfup}`

If you define your own \LaTeX{} or MathJax macros for these commands, set `html` to `FALSE`.

Microsoft Office and LibreOffice output

If `word_proc` is `TRUE` and you render to Word or ODT (or related formats such as PowerPoint), the following \LaTeX{} math commands are defined:

- `\textit{symit}`
- `\textup{symup}`
- `\textbf{symbfit}`
- `\textbf{symbfup}`

If you define your own \LaTeX{}, Office, or OpenDocument macros for these commands, set `word_proc` to `FALSE`.

See Also

Other output functions: `generate_bib()`, `metabulate()`

Examples

```r
## Include this line as 'asis' output in your RMarkdown document:
metabulate_rmd_helper()

## If you've already included `\usepackage{unicode-math}` in your RMarkdown header
## for PDF (and related formats) header, set `latex` to `FALSE`:
metabulate_rmd_helper(latex = FALSE)
```
metareg

Compute meta-regressions

Description

This function is a wrapper for `metafor`'s `rma` function that computes meta-regressions for all bare-bones and individual-correction meta-analyses within an object. It makes use of both categorical and continuous moderator information stored in the meta-analysis object and allows for interaction effects to be included in the regression model. Output from this function will be added to the meta-analysis object in a list called `follow_up_analyses`. If using this function with a multi-construct meta-analysis object from `ma_r` or `ma_d`, note that the `follow_up_analyses` list is appended to the meta-analysis object belonging to a specific construct pair within the `construct_pairs` list.

Usage

```r
metareg(ma_obj, formula_list = NULL, ...)
```

Arguments

- `ma_obj` Meta-analysis object.
- `formula_list` Optional list of regression formulas to evaluate. NOTE: If there are spaces in your moderator names, replace them with underscores (i.e., "_") so that the formula(s) will perform properly. The function will remove spaces in the data, you only have to account for this in `formula_list` when you supply your own formula(s).
- `...` Additional arguments.

Value

`ma_obj` with meta-regression results added (see `ma_obj$follow_up_analyses$metareg`).

Examples

```r
## Meta-analyze the data from Gonzalez-Mule et al. (2014)
## Note: These are corrected data and we have confirmed with the author that
## these results are accurate:
ma_obj <- ma_r_ic(rxyi = rxyi, n = n, hs_override = TRUE, data = data_r_gonzalezmule_2014,
                  rxx = rxxi, ryy = ryyi, ux = ux, indirect_rr_x = TRUE,
                  correct_rr_x = TRUE, moderators = Complexity)

## Pass the meta-analysis object to the meta-regression function:
ma_obj <- metareg(ma_obj)

## Examine the meta-regression results for the bare-bones and corrected data:
ma_obj$metareg[[1]]$barebones$'Main Effects'
ma_obj$metareg[[1]]$individual_correction$true_score$'Main Effects'
```
## Meta-analyze simulated d-value data

dat <- data_d_meas_multi
## Simulate a random moderator
set.seed(100)
dat$moderator <- sample(1:2, nrow(dat), replace = TRUE)
ma_obj <- ma_d(ma_method = "ic", d = d, n1 = n1, n2 = n2, ryy = ryyi,
               construct_y = construct, sample_id = sample_id,
               moderators = moderator, data = dat)

## Pass the meta-analysis object to the meta-regression function:
ma_obj <- metareg(ma_obj)

## Examine the meta-regression results for the bare-bones and corrected data:
ma_obj$metareg[[1]]$barebones$'Main Effects'
ma_obj$metareg[[1]]$individual_correction$latentGroup_latentY$'Main Effects'

### mix_dist

**Descriptive statistics for a mixture distribution**

#### Description

Compute descriptive statistics for a mixture distribution. This function returns the grand mean, the pooled sample variance (mean square within), variance of sample means (mean square between), portions of the total variance that are within and between groups, and mixture (total sample) variance of the mixture sample data.

#### Usage

```r
mix_dist(mean_vec, var_vec, n_vec, unbiased = TRUE, na.rm = FALSE)
```

#### Arguments

- `mean_vec`: Vector of sample means.
- `var_vec`: Vector of sample variances.
- `n_vec`: Vector of sample sizes.
- `unbiased`: Logical scalar determining whether variance should be unbiased (TRUE; default) or maximum-likelihood (FALSE).
- `na.rm`: Logical scalar determining whether to remove missing values prior to computing output (TRUE) or not (FALSE; default)

#### Details

The grand mean of a mixture distribution is computed as:

$$
\mu = \frac{\sum_{i=1}^{k} \bar{x}_i n_i}{\sum_{i=1}^{k} n_i}
$$
where $\mu$ is the grand mean, $\bar{x}_i$ represents the sample means, and $n_i$ represents the sample sizes.

Maximum-likelihood mixture variances are computed as:

\[
\begin{align*}
\text{var}_\text{pooled}_{ML} &= MSW_{ML} = \frac{\sum_{i=1}^{k} (\bar{x}_i - \mu) n_i}{\sum_{i=1}^{k} n_i} \\
\text{var}_\text{means}_{ML} &= MSB_{ML} = \frac{\sum_{i=1}^{k} (\bar{x}_i - \mu) n_i}{k} \\
\text{var}_B G_{ML} &= \frac{\sum_{i=1}^{k} (\bar{x}_i - \mu) n_i}{\sum_{i=1}^{k} n_i} \\
\text{var}_W G_{ML} &= \frac{\sum_{i=1}^{k} v_i n_i}{\sum_{i=1}^{k} n_i} \\
\text{var}_\text{mix}_{ML} &= \text{var}_B G_{ML} + \text{var}_W G_{ML}
\end{align*}
\]

where $v_i$ represents the sample variances.

Unbiased mixture variances are computed as:

\[
\begin{align*}
\text{var}_\text{pooled}_{Unbiased} &= MSW_{Unbiased} = \frac{\sum_{i=1}^{k} v_i (n_i - 1)}{(\sum_{i=1}^{k} n_i) - k} \\
\text{var}_\text{means}_{Unbiased} &= MSB_{Unbiased} = \frac{\sum_{i=1}^{k} (\bar{x}_i - \mu) n_i}{k - 1} \\
\text{var}_B G_{Unbiased} &= \frac{\sum_{i=1}^{k} (\bar{x}_i - \mu) n_i}{(\sum_{i=1}^{k} n_i) - 1} \\
\text{var}_W G_{Unbiased} &= \frac{\sum_{i=1}^{k} v_i (n_i - 1)}{(\sum_{i=1}^{k} n_i) - 1} \\
\text{var}_\text{mix}_{Unbiased} &= \text{var}_B G_{Unbiased} + \text{var}_W G_{Unbiased}
\end{align*}
\]

Value

The mean, pooled sample (within-sample) variance, variance of sample means (between-groups), and mixture (total sample) variance of the mixture sample data.

Examples

\[
mix\_dist(\text{mean}\_\text{vec} = c(-.5, 0, .5), \text{var}\_\text{vec} = c(.9, 1, 1.1), \text{n}\_\text{vec} = c(100, 100, 100))
\]
mix_matrix

Estimate mixture covariance matrix from within-group covariance matrices

Description

Estimate mixture covariance matrix from within-group covariance matrices

Usage

mix_matrix(
  sigma_list,
  mu_mat,
  p_vec,
  N = Inf,
  group_names = NULL,
  var_names = NULL
)

Arguments

sigma_list  List of covariance matrices.
mu_mat  Matrix of mean parameters, with groups on the rows and variables on the columns.
p_vec  Vector of proportion of cases in each group.
N  Optional total sample size across all groups (used to compute unbiased covariance estimates).
group_names  Optional vector of group names.
var_names  Optional vector of variable names.

Value

List of mixture covariances and means.

Examples

out <- unmix_matrix(sigma_mat = reshape_vec2mat(.5, order = 2),
  mu_mat = rbind(c(0, 0), c(.5, 1)),
  p_vec = c(.3, .7), N = 100)

mix_matrix(sigma_list = out$cov_group_unbiased,
  mu_mat = out$means_raw[-3,],
  p_vec = out$p_group, N = out$N)
mix_r_2group  Estimate the mixture correlation for two groups

Description

Estimate the mixture correlation for two groups.

Usage

\[
\text{mix}_r_2\text{group}(rxy, dx, dy, p = 0.5)
\]

Arguments

- **rxy**: Average within-group correlation
- **dx**: Standardized mean difference between groups on X.
- **dy**: Standardized mean difference between groups on Y.
- **p**: Proportion of cases in one of the two groups.

Details

The average within-group correlation is estimated as:

\[
\rho_{xyWG} = \rho_{xyMix} \sqrt{\left(d_x^2 p(1 - p) + 1\right) \left(d_y^2 p(1 - p) + 1\right) - \sqrt{d_x^2 d_y^2 p^2(1 - p)^2}}
\]

where \(\rho_{xyWG}\) is the average within-group correlation, \(\rho_{xyMix}\) is the overall mixture correlation, \(d_x\) is the standardized mean difference between groups on X, \(d_y\) is the standardized mean difference between groups on Y, and \(p\) is the proportion of cases in one of the two groups.

Value

A vector of two-group mixture correlations

Examples

\[
\text{mix}_r_2\text{group}(rxy = .375, dx = 1, dy = 1, p = .5)
\]
plot_forest

Create forest plots

Description

Create forest plots

Usage

plot_forest(
  ma_obj, 
  analyses = "all", 
  match = c("all", "any"), 
  case_sensitive = TRUE, 
  show_filtered = FALSE, 
  ma_facetname = "Summary", 
  facet_levels = NULL, 
  conf_level = NULL, 
  conf_method = NULL, 
  x_limits = NULL, 
  x_breaks = NULL, 
  x_lab = NULL, 
  y_lab = "Reference"
)

Arguments

ma_obj  Meta-analysis object.
analyses  Which analyses to extract? Can be either "all" to extract references for all meta-analyses in the object (default) or a list containing arguments for filter_ma().
match  Should extracted meta-analyses match all (default) or any of the criteria given in analyses?
case_sensitive  Logical scalar that determines whether character values supplied in analyses should be treated as case sensitive (TRUE, default) or not (FALSE).
show_filtered  Logical scalar that determines whether the meta-analysis object given in the output should be the modified input object (FALSE, default) or the filtered object (TRUE).
ma_facetname  Label to use for meta-analysis results in the ggplot2::facet_grid() function.
facet_levels  Order in which moderator levels should be displayed.
conf_level  Confidence level to define the width of the confidence interval. If NULL (default), uses the level set when ma_obj was estimated.
conf_method  Distribution to be used to compute confidence intervals (either "t" for t distribution or "norm" for normal distribution). If NULL (default), uses the method set when ma_obj was estimated.
## plot_funnel

Create funnel plots

### Description

This function creates funnel plots for meta-analyses (plots of effect size versus standard error).

### Usage

```r
plot_funnel(
  ma_obj,
  se_type = c("auto", "mean", "sample"),
  label_es = NULL,
  conf_level = c(0.95, 0.99),
  conf_linetype = c("dashed", "dotted"),
)```

### Value

A list of forest plots.

### Author(s)

Based on code by John Sakaluk

### Examples

```r
## Not run:
ma_obj <- ma_r(ma_method = "ic", rxyi = rxyi, n = n, rxx = rxxi, ryy = ryyi,
               construct_x = x_name, construct_y = y_name, sample_id = sample_id,
               moderators = moderator, data = data_r_meas_multi)
plot_forest(ma_obj = ma_obj)
plot_forest(ma_obj = ma_obj, analyses = list(pair_id = 2))
plot_forest(ma_obj = ma_obj, analyses = list(pair_id = 1), show_filtered = TRUE)

## d values
ma_obj <- ma_d(ma_method = "ic", d = d, n1 = n1, n2 = n2, ryy = ryyi,
               construct_y = construct, sample_id = sample_id,
               data = data_d_meas_multi)
plot_forest(ma_obj = ma_obj)
plot_forest(ma_obj = ma_obj, analyses = list(pair_id = 2))
plot_forest(ma_obj = ma_obj, analyses = list(pair_id = 1, analysis_id = 1), show_filtered = TRUE)

## End(Not run)
```
conf_fill = NA,
conf_alpha = 1,
null_effect = NA,
null_conf_level = c(0.9, 0.95, 0.99),
null_conf_linetype = c("solid", "dashed", "dotted"),
null_conf_fill = "black",
null_conf_alpha = c(0.1, 0.2, 0.4),
analyses = "all",
match = c("all", "any"),
case_sensitive = TRUE,
show_filtered = FALSE
)

plot_cefp(
  ma_obj,
  se_type = "sample",
  label_es = NULL,
  conf_level = NA,
  conf_linetype = NA,
  conf_fill = NA,
  conf_alpha = 1,
  null_effect = NULL,
  null_conf_level = c(0.9, 0.95, 0.99),
  null_conf_linetype = c("solid", "dashed", "dotted"),
  null_conf_fill = "black",
  null_conf_alpha = c(0, 0.2, 0.4),
analyses = "all",
  match = c("all", "any"),
  case_sensitive = TRUE,
  show_filtered = FALSE
)

**Arguments**

- **ma_obj**
  Meta-analysis object.

- **se_type**
  Method to calculate standard errors (y-axis). Options are "auto" (default) to use the same method as used to estimate the meta-analysis models, "mean" to calculate SEs using the mean effect size and individual sample sizes, or "sample" to use the SE calculated using the sample effect sizes and sample sizes.

- **label_es**
  Label for effect size (x-axis). Defaults to "Correlation (r)" for correlation meta-analyses, "Cohen’s d (Hedges’s g)" for d value meta-analyses, and "Effect size" for generic meta-analyses.

- **conf_level**
  Confidence regions levels to be plotted (default: .95, .99).

- **conf_linetype**
  Line types for confidence region boundaries. Length should be either 1 or equal to the length of conf_level.

- **conf_fill**
  Colors for confidence regions. Set to NA for transparent. Length should be either 1 or equal to the length of conf_level.
conf_alpha  Transparency level for confidence regions. Length should be either 1 or equal to the length of conf_level.
null_effect  Null effect to be plotted for contour-enhanced funnel plots. If NA, not shown. If NULL, set to the null value for the effect size metric (0 for correlations and d values).
null_conf_level  Null-effect confidence regions levels to be plotted (default: .90, .95, .99).
null_conf_linetype  Line types for null-effect confidence region boundaries. Length should be either 1 or equal to the length of null_conf_level.
null_conf_fill  Colors for null-effect confidence regions. Set to NA for transparent. Length should be either 1 or equal to the length of null_conf_level.
null_conf_alpha  Transparency level for null-effect confidence regions. Length should be either 1 or equal to the length of null_conf_level.
analyses  Which analyses to extract? Can be either "all" to extract references for all meta-analyses in the object (default) or a list containing arguments for filter_ma.
match  Should extracted meta-analyses match all (default) or any of the criteria given in analyses?
case_sensitive  Logical scalar that determines whether character values supplied in analyses should be treated as case sensitive (TRUE, default) or not (FALSE).
show_filtered  Logical scalar that determines whether the meta-analysis object given in the output should be the modified input object (FALSE, default) or the filtered object (TRUE).

Details

Both traditional funnel plots and contour-enhanced funnel plots are provided. Contour-enhanced funnel plots show comparison regions for varying null-hypothesis significance test levels and can be useful for detecting publication bias.

Value

A list of funnel plots.

Author(s)

Based on code by John Sakaluk

Examples

```r
## Not run:
## Correlations
ma_obj <- ma_r(ma_method = "ic", rxyi = rxyi, n = n, rxx = rxxi, ryy = ryyi,
construct_x = x_name, construct_y = y_name, sample_id = sample_id,
moderators = moderator, data = data_r_meas_multi)
plot_funnel(ma_obj = ma_obj)
```
```
plot_funnel(ma_obj = ma_obj, analyses = list(pair_id = 2))
plot_funnel(ma_obj = ma_obj, analyses = list(pair_id = 1, analysis_id = 1), show_filtered = TRUE)

## d values
ma_obj <- ma_d(ma_method = "ic", d = d, n1 = n1, n2 = n2, ryy = ryyi,
              construct_y = construct, sample_id = sample_id,
              data = data_d_meas_multi)
plot_funnel(ma_obj = ma_obj)
plot_funnel(ma_obj = ma_obj, analyses = list(pair_id = 2))
plot_funnel(ma_obj = ma_obj, analyses = list(pair_id = 1, analysis_id = 1), show_filtered = TRUE)

## End(Not run)
```

---

**predict**  
*Prediction method for objects of classes deriving from "lm_mat"*

**Description**  
Prediction method for objects of classes deriving from "lm_mat"

**Arguments**
- **object**: Object of class inheriting from "lm_mat"
- **newdata**: An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
- **se.fit**: A switch indicating if standard errors are required.
- **df**: Degrees of freedom for scale.
- **interval**: Type of interval calculation. Can be abbreviated.
- **level**: Tolerance/confidence level.
- **...**: further arguments passed to or from other methods.

**Value**  
An set of predicted values

---

**print**  
*Print methods for psychmeta*

**Description**  
Print methods for psychmeta output objects with classes exported from psychmeta.
### Arguments

- **x**: Object to be printed (object is used to select a method).
- **...**: Additional arguments.
- **digits**: Number of digits to which results should be rounded.
- **ma_methods**: Meta-analytic methods to be included. Valid options are: "bb", "ic", and "ad".
- **correction_types**: Types of meta-analytic corrections to be included. Valid options are: "ts", "vgx", and "vgy".
- **verbose**: Logical scalar that determines whether printed object should contain verbose information (e.g., non-standard columns of meta-analytic output; TRUE) or not (FALSE).
- **n**: For `print.ma_psychmeta()` and `print.ad_tibble()`, number of rows to print for tibble. Defaults to all rows. See `tibble::print.tbl()` for details.
- **width**: For `print.ma_psychmeta()` and `print.ad_tibble()`, width of text output to generate for tibble. See `tibble::print.tbl()` for details.
- **n_extra**: For `print.ma_psychmeta()` and `print.ad_tibble()`, number of extra columns to print abbreviated information for, if the width is too small for the entire meta-analysis tibble. See `tibble::print.tbl()` for details.
- **symbolic.cor**: For `print.lm_mat()`, Logical. If TRUE, print the correlations in a symbolic form (see `stats::symnum()`) rather than as numbers.
- **signif.stars**: For `print.lm_mat()`, Logical. If TRUE, 'significance stars' are printed for each coefficient.

---

## Description

Copy class and attributes from the original version of an object to a modified version.

## Usage

```r
reattribute(x, result)
```

## Arguments

- **x**: The original object, which has a class/attributes to copy
- **result**: The modified object, which is / might be missing the class/attributes.

## Value

`result`, now with class/attributes restored.
**reshape_mat2dat**

*Extract a long-format correlation database from a correlation matrix and its supporting vectors/matrices of variable information*

**Description**

This function is designed to extract data from a correlation matrix that is in the format commonly published in journals, with leading columns of construct names and descriptive statistics being listed along with correlation data.

**Usage**

```r
reshape_mat2dat(
  var_names,
  cor_data,
  common_data = NULL,
  unique_data = NULL,
  diag_label = NULL,
  lower_tri = TRUE,
  data = NULL
)
```

**Arguments**

- `var_names`: Vector (or scalar column name to match with `data`) containing variable names.
- `cor_data`: Square matrix (or vector of column names to match with `data`) containing correlations among variables.
- `common_data`: Vector or matrix (or vector of column names to match with `data`) of data common to both X and Y variables (e.g., sample size, study-wise moderators).
- `unique_data`: Vector or matrix (or vector of column names to match with `data`) of data unique to X and Y variables (e.g., mean, SD, reliability).
- `diag_label`: Optional name to attribute to values extracted from the diagonal of the matrix (if `NULL`, no values are extracted from the diagonal).
- `lower_tri`: Logical scalar that identifies whether the correlations are in the lower triangle (`TRUE`) or in the upper triangle `FALSE` of the matrix.
- `data`: Matrix or data frame containing study data (when present, column names of `data` will be matched to column names provided as other arguments).

**Value**

Long-format data frame of correlation data, variable names, and supporting information

**Author(s)**

Jack W. Kostal
Examples

## Create a hypothetical matrix of data from a small study:
mat <- data.frame(var_names = c("X", "Y", "Z"),
  n = c(100, 100, 100),
  mean = c(4, 5, 3),
  sd = c(2.4, 2.6, 2),
  rel = c(.8, .7, .85),
  reshape_vec2mat(cov = c(.3, .4, .5)))

## Arguments can be provided as quoted characters or as the unquoted names of `data`'s columns:
reshape_mat2dat(var_names = var_names,
  cor_data = c("Var1", "Var2", "Var3"),
  common_data = "n",
  unique_data = c("mean", "sd", "rel"),
  data = mat)

## Arguments can also provided as raw vectors, matrices, or data frames, without a data argument:
reshape_mat2dat(var_names = mat[,1],
  cor_data = mat[,6:8],
  common_data = mat[,2],
  unique_data = mat[,3:5])

## If data is not null, arguments can be a mix of matrix/data frame/vector and column-name arguments
reshape_mat2dat(var_names = mat[,1],
  cor_data = mat[,6:8],
  common_data = "n",
  unique_data = c("mean", "sd", "rel"),
  data = mat)

---

**reshape_vec2mat**  
Assemble a variance-covariance matrix

Description

The **reshape_vec2mat** function facilitates the creation of square correlation/covariance matrices from scalars or vectors of variances/covariances. It allows the user to supply a vector of covariances that make up the lower triangle of a matrix, determines the order of the matrix necessary to hold those covariances, and constructs a matrix accordingly.

Usage

```r
reshape_vec2mat(
  cov = NULL,
  var = NULL,
  order = NULL,
  var_names = NULL,
  by_row = FALSE,
  diag = FALSE
)
```
Arguments

cov  Scalar or vector of covariance information to include the lower-triangle positions of the matrix (default value is zero). If a vector, the elements must be provided in the order associated with concatenated column (by_row = FALSE; default) or row (by_row = TRUE) vectors of the lower triangle of the desired matrix. If variances are included in these values, set the diag argument to TRUE.

var  Scalar or vector of variance information to include the diagonal positions of the matrix (default value is 1).

order  If cov and var are scalars, this argument determines the number of variables to create in the output matrix.

var_names  Optional vector of variable names.

by_row  Logical scalar indicating whether cov values should fill the lower triangle by row (TRUE) or by column (FALSE; default).

diag  Logical scalar indicating whether cov values include variances (FALSE by default; if TRUE, the variance values supplied with the cov argument will supersede the var argument).

Value

A variance-covariance matrix

Examples

```r
## Specify the lower triangle covariances
## Can provide names for the variables
reshape_vec2mat(cov = c(.3, .2, .4), var_names = c("x", "y", "z"))

## Specify scalar values to repeat for the covariances and variances
reshape_vec2mat(cov = .3, var = 2, order = 3)

## Give a vector of variances to create a diagonal matrix
reshape_vec2mat(var = 1:5)

## Specify order only to create identity matrix
reshape_vec2mat(order = 3)

## Specify order and scalar variance to create a scalar matrix
reshape_vec2mat(var = 2, order = 3)

## A quick way to make a 2x2 matrix for bivariate correlations
reshape_vec2mat(cov = .2)
```
Reshape database from wide format to long format

Description

This function automates the process of converting a wide-format database (i.e., a database in which intercorrelations between construct pairs define the columns, such that there are multiple columns of correlations) to a long-format database (i.e., a database with just one column of correlations). The meta-analysis functions in psychmeta work best with long-format databases, so this function can be a helpful addition to one’s workflow when data are organized in a wide format.

Usage

```r
reshape_wide2long(
  data,
  common_vars = NULL,
  es_design = NULL,
  n_design = NULL,
  other_design = NULL,
  es_name = "rxyi",
  missing_col_action = c("warn", "ignore", "stop")
)
```

Arguments

- **data** Database of data for use in a meta-analysis in "wide" format.
- **common_vars** String vector of column names relevant to all variables in data.
- **es_design** p x p matrix containing the names of columns of intercorrelations among variables in the lower triangle of the matrix.
- **n_design** Scalar sample-size column name or a p x p matrix containing the names of columns of sample sizes the lower triangle of the matrix.
- **other_design** A matrix with variable names on the rows and names of long-format variables to create on the columns. Elements of this matrix must be column names of data.
- **es_name** Name of the effect size represented in data.
- **missing_col_action** Character scalar indicating how missing columns should be handled. Options are: "warn", "ignore", and "stop"

Value

A long-format database
Examples

```r
n_params = c(mean = 150, sd = 20)
rho_params <- list(c(.1, .3, .5),
                   c(mean = .3, sd = .05),
                   rbind(value = c(.1, .3, .5), weight = c(1, 2, 1)))
rel_params = list(c(.7, .8, .9),
                  c(mean = .8, sd = .05),
                  rbind(value = c(.7, .8, .9), weight = c(1, 2, 1)))
sr_params = c(.1, .3, .5)
rel_params = list(c(.5, .6, .7))
wtk_params = list(c(1, 2, 3),
                  c(mean = 2, sd = .25),
                  rbind(value = c(1, 2, 3), weight = c(1, 2, 1)))
```

```r
## Simulate with wide format
## Not run:
data <- simulate_r_database(k = 10, n_params = n_params, rho_params = rho_params,
                            rel_params = rel_params, sr_params = sr_params,
                            sr_composite_params = sr_composite_params, wt_params = wt_params,
                            var_names = c("X", "Y", "Z"), format = "wide")$statistics
## End(Not run)
## Define values to abstract from the data object
common_vars <- "sample_id"
es_design <- matrix(NA, 3, 3)
var_names <- c("X", "Y", "Z")
es_design[lower.tri(es_design)] <- c("rxyi_X_Y", "rxyi_X_Z", "rxyi_Y_Z")
rownames(es_design) <- colnames(es_design) <- var_names
n_design <- "ni"
other_design <- cbind(rxxi = paste0("parallel_rxxi_", var_names),
                      ux_local = paste0("ux_local_", var_names),
                      ux_external = paste0("ux_external_", var_names))
rownames(other_design) <- var_names
```

```
## Reshape the data to "long" format
reshape_wide2long(data = data, common_vars = common_vars, es_design = es_design,
                  n_design = n_design, other_design = other_design)
```

---

**Sensitivity analyses for meta-analyses**

**Description**

Wrapper function to compute bootstrap analyses, leave-one-out analyses, and cumulative meta-analyses. This function helps researchers to examine the stability/fragility of their meta-analytic results with bootstrapping and leave-one-out analyses, as well as detect initial evidence of publication bias with cumulative meta-analyses.
sensitivity

Usage

sensitivity(
    ma_obj,
    leave1out = TRUE,
    bootstrap = TRUE,
    cumulative = TRUE,
    sort_method = c("weight", "n", "inv_var"),
    boot_iter = 1000,
    boot_conf_level = 0.95,
    boot_ci_type = c("bca", "norm", "basic", "stud", "perc"),
    ... )

sensitivity_bootstrap(
    ma_obj,
    boot_iter = 1000,
    boot_conf_level = 0.95,
    boot_ci_type = c("bca", "norm", "basic", "stud", "perc"),
    ... )

sensitivity_cumulative(ma_obj, sort_method = c("weight", "n", "inv_var"), ...)

sensitivity_leave1out(ma_obj, ...)

Arguments

ma_obj  Meta-analysis object.
leave1out Logical scalar determining whether to compute leave-one-out analyses (TRUE)
or not (FALSE).
bootstrap Logical scalar determining whether bootstrapping is to be performed (TRUE) or
not (FALSE).
cumulative Logical scalar determining whether a cumulative meta-analysis is to be com-
puted (TRUE) or not (FALSE).
sort_method Method to sort samples in the cumulative meta-analysis. Options are "weight"
to sort by weight (default), "n" to sort by sample size, and "inv_var" to sort by
inverse variance.
boot_iter Number of bootstrap iterations to be computed.
boot_conf_level Width of confidence intervals to be constructed for all bootstrapped statistics.
boot_ci_type Type of bootstrapped confidence interval. Options are "bca", "norm", "basic",
"stud", and "perc" (these are "type" options from the boot::boot.ci function).
Default is "bca". Note: If you have too few iterations, the "bca" method will
not work and you will need to either increase the iterations or choose a different
method.
... Additional arguments.
Value

An updated meta-analysis object with sensitivity analyses added.

• When bootstrapping is performed, the bootstrap section of the follow_up_analyses section of the updated ma_obj returned by this function will contain both a matrix summarizing the mean, variance, and confidence intervals of the bootstrapped samples and a table of meta-analytic results from all bootstrapped samples.

• When leave-one-out analyses are performed, the ma_obj will acquire a list of leave-one-out results in its follow_up_analyses section that contains a table of all leave-one-out meta-analyses along with plots of the mean and residual variance of the effect sizes in the meta-analyses.

• When cumulative meta-analysis is performed, the ma_obj will acquire a list of cumulative meta-analysis results in its follow_up_analyses section that contains a table of all meta-analyses computed along with plots of the mean and residual variance of the effect sizes in the meta-analyses, sorted by the order in which studies were added to the meta-analysis.

Examples

## Not run:
## Run a meta-analysis using simulated correlation data:
ma_obj <- ma_r_ic(rxyi = rxyi, n = n, rxx = rxxi, ryy = ryyi, ux = ux,
  correct_rr_y = FALSE, data = data_r_uvirr)
ma_obj <- ma_r_ad(ma_obj, correct_rr_y = FALSE)
## Pass the meta-analysis object to the sensitivity() function:
ma_obj <- sensitivity(ma_obj = ma_obj, boot_iter = 10,
  boot_ci_type = "norm", sort_method = "inv_var")

## Examine the tables and plots produced for the IC meta-analysis:
ma_obj$bootstrap[[1]]$barebones
ma_obj$bootstrap[[1]]$individual_correction$true_score
ma_obj$leave1out[[1]]$individual_correction$true_score
ma_obj$cumulative[[1]]$individual_correction$true_score

## Examine the tables and plots produced for the AD meta-analysis:
ma_obj$bootstrap[[1]]$artifact_distribution$true_score
ma_obj$leave1out[[1]]$artifact_distribution$true_score
ma_obj$cumulative[[1]]$artifact_distribution$true_score

## Run a meta-analysis using simulated d-value data:
ma_obj <- ma_d_ic(d = d, n1 = n1, n2 = n2, ryy = ryyi,
  data = filter(data_d_meas_multi, construct == "Y"))
ma_obj <- ma_d_ad(ma_obj)
## Pass the meta-analysis object to the sensitivity() function:
ma_obj <- sensitivity(ma_obj = ma_obj, boot_iter = 10,
  boot_ci_type = "norm", sort_method = "inv_var")

## Examine the tables and plots produced for the IC meta-analysis:
ma_obj$bootstrap[[1]]$barebones
simulate_alpha

Generate a vector of simulated sample alpha coefficients

**Description**

This function generates inter-item covariance matrices from a population matrix and computes a coefficient alpha reliability estimate for each matrix.

**Usage**

```r
simulate_alpha(
  item_mat = NULL,
  alpha = NULL,
  k_items = NULL,
  n_cases,
  k_samples,
  standarized = FALSE
)
```

**Arguments**

- **item_mat**: Item correlation/covariance matrix. If `item_mat` is not supplied, the user must supply both `alpha` and `k_items`. If `item_mat` is `NULL`, the program will assume that all item intercorrelations are equal.
- **alpha**: Population alpha value. Must be supplied if `item_mat` is `NULL`.
- **k_items**: Number of items on the test to be simulated. Must be supplied if `item_mat` is `NULL`.
- **n_cases**: Number of cases to simulate in sampling distribution of alpha.
- **k_samples**: Number of samples to simulate.
- **standarized**: Should alpha be computed from correlation matrices (TRUE) or unstandardized covariance matrices (FALSE)?

**Value**

A vector of simulated sample alpha coefficients
Examples

```r
## Define a hypothetical matrix:
item_mat <- reshape_vec2mat(cov = .3, order = 12)

## Simulations of unstandardized alphas
set.seed(100)
simulate_alpha(item_mat = item_mat, n_cases = 50, k_samples = 10, standarized = FALSE)
set.seed(100)
simulate_alpha(alpha = mean(item_mat[lower.tri(item_mat)]) / mean(item_mat),
               k_items = ncol(item_mat), n_cases = 50, k_samples = 10, standarized = FALSE)

## Simulations of standardized alphas
set.seed(100)
simulate_alpha(item_mat = item_mat, n_cases = 50, k_samples = 10, standarized = TRUE)
set.seed(100)
simulate_alpha(alpha = mean(item_mat[lower.tri(item_mat)]) / mean(item_mat),
               k_items = ncol(item_mat), n_cases = 50, k_samples = 10, standarized = TRUE)
```

Description

The `simulate_d_database` function generates databases of psychometric d value data from sample-size parameters, correlation parameters, mean parameters, standard deviation parameters, reliability parameters, and selection-ratio parameters. The output database can be provided in a long format. If composite variables are to be formed, parameters can also be defined for the weights used to form the composites as well as the selection ratios applied to the composites. This function will return a database of statistics as well as a database of parameters - the parameter database contains the actual study parameters for each simulated sample (without sampling error) to allow comparisons between meta-analytic results computed from the statistics and the actual means and variances of parameters. The `merge_simdat_d` function can be used to merge multiple simulated databases and the `sparsify_simdat_d` function can be used to randomly delete artifact information (a procedure commonly done in simulations of artifact-distribution methods).

Usage

```r
simulate_d_database(
  k,
  n_params,
  rho_params,
  mu_params = NULL,
  sigma_params = 1,
  rel_params = 1,
  sr_params = 1,
  k_items_params = 1,
  wt_params = NULL,
  allow_neg_wt = FALSE,
)```
simulate_d_database

```r
sr_composite_params = NULL,
group_names = NULL,
var_names = NULL,
composite_names = NULL,
diffs_as_obs = FALSE,
show_applicant = FALSE,
keep_vars = NULL,
decimals = 2,
max_iter = 100,
...
)
```

**Arguments**

- **k**
  - Number of studies to simulate.

- **n_params**
  - List of parameter distributions (or data-generation function; see details) for subgroup sample sizes.

- **rho_params**
  - List containing a list of parameter distributions (or data-generation functions; see details) for correlations for each simulated group. If simulating data from a single fixed population matrix in each group, supply a list of those matrices for this argument (if the diagonals contain non-unity values and `sigma_params` argument is not specified, those values will be used as variances).

- **mu_params**
  - List containing a list of parameter distributions (or data-generation functions; see details) for means for each simulated group. If NULL, all means will be set to zero.

- **sigma_params**
  - List containing a list of parameter distributions (or data-generation functions; see details) for standard deviations for each simulated group. If NULL, all standard deviations will be set to unity.

- **rel_params**
  - List containing a list of parameter distributions (or data-generation functions; see details) for reliabilities for each simulated group. If NULL, all reliabilities will be set to unity.

- **sr_params**
  - List of parameter distributions (or data-generation functions; see details) for selection ratios. If NULL, all selection ratios will be set to unity.

- **k_items_params**
  - List of parameter distributions (or data-generation functions; see details) for the number of test items comprising each of the variables to be simulated (all are single-item variables by default).

- **wt_params**
  - List of parameter distributions (or data-generation functions; see details) to create weights for use in forming composites. If multiple composites are formed, the list should be a list of lists, with the general format: `list(comp1_params = list(...params...), comp2_params = list(...params...), etc.)`.

- **allow_neg_wt**
  - Logical scalar that determines whether negative weights should be allowed (TRUE) or not (FALSE).

- **sr_composite_params**
  - Parameter distributions (or data-generation functions; see details) for composite selection ratios.

- **group_names**
  - Optional vector of group names.
simulate_d_database

var_names
Optional vector of variable names for all non-composite variables.

composite_names
Optional vector of names for composite variables.

diffs_as_obs
Logical scalar that determines whether standard deviation parameters represent standard deviations of observed scores (TRUE) or of true scores (FALSE; default).

show_applicant
Should applicant data be shown for sample statistics (TRUE) or suppressed (FALSE)?

keep_vars
Optional vector of variable names to be extracted from the simulation and returned in the output object. All variables are returned by default. Use this argument when only some variables are of interest and others are generated solely to serve as selection variables.

decimals
Number of decimals to which statistical results (not parameters) should be rounded. Rounding to 2 decimal places best captures the precision of data available from published primary research.

max_iter
Maximum number of iterations to allow in the parameter selection process before terminating with convergence failure. Must be finite.

... Additional arguments.

Details

Values supplied as any argument with the suffix "params" can take any of three forms (see Examples for a demonstration of usage):

- A vector of values from which study parameters should be sampled.
- A vector containing a mean with a variance or standard deviation. These values must be named "mean," "var," and "sd", respectively, for the program to recognize which value is which.
- A matrix containing a row of values (this row must be named "values") from which study parameters should be sampled and a row of weights (this row must be labeled 'weights') associated with the values to be sampled.
- A matrix containing a column of values (this column must be named "values") from which study parameters should be sampled and a column of weights (this column must be labeled 'weights') associated with the values to be sampled.
- A function that is configured to generate data using only one argument that defines the number of cases to generate, e.g., fun(n = 10).

Value

A database of simulated primary studies’ statistics and analytically determined parameter values.

Examples

```r
if (requireNamespace("nor1mix", quietly = TRUE)) {
    ## Define sample sizes, means, and other parameters for each of two groups:
    n_params <- list(c(mean = 200, sd = 20),
                     c(mean = 100, sd = 20))
    rho_params <- list(list(c(.3, .4, .5)),
                        list(c(.3, .4, .5)))
    mu_params <- list(list(c(mean = .5, sd = .5), c(-.5, 0, .5)),
                        list(c(mean = .5, sd = .5), c(-.5, 0, .5)))
```
simulate_d_sample

```r
list(c(mean = 0, sd = .5), c(-.2, 0, .2))
sigma_params <- list(list(1, 1), 
                     list(1, 1))
rel_params <- list(list(.8, .8),
                   list(.8, .8))
sr_params <- list(1, .5)

simulate_d_database(k = 5, n_params = n_params, rho_params = rho_params,
                     mu_params = mu_params, sigma_params = sigma_params,
                     rel_params = rel_params, sr_params = sr_params,
                     k_items = c(4, 4),
                     group_names = NULL, var_names = c("y1", "y2"),
                     show_applicant = TRUE, keep_vars = c("y1", "y2"), decimals = 2)
```

---

**simulate_d_sample**  
*Simulate a sample of psychometric d value data with measurement error, direct range restriction, and/or indirect range restriction*

---

**Description**

This function generates a simulated psychometric sample consisting of any number of groups and computes the $d$ values that result after introducing measurement error and/or range restriction.

**Usage**

```r
simulate_d_sample(
  n_vec,
  rho_mat_list,
  mu_mat,
  sigma_mat = 1,
  rel_mat = 1,
  sr_vec = 1,
  k_items_vec = 1,
  wt_mat = NULL,
  sr_composites = NULL,
  group_names = NULL,
  var_names = NULL,
  composite_names = NULL,
  diffs_as_obs = FALSE
)
```

**Arguments**

- `n_vec`: Vector of sample sizes (or a vector of proportions, if parameters are to be estimated).
- `rho_mat_list`: List of true-score correlation matrices.
- `mu_mat`: Matrix of mean parameters, with groups on the rows and variables on the columns.
simulate_matrix

sigma_mat Matrix of standard-deviation parameters, with groups on the rows and variables on the columns.
rel_mat Matrix of reliability parameters, with groups on the rows and variables on the columns.
sr_vec Vector of selection ratios.
k_items_vec Number of test items comprising each of the variables to be simulated (all are single-item variables by default).
wt_mat Optional matrix of weights to use in forming a composite of the variables in rho_mat. Matrix should have as many rows (or vector elements) as there are variables in rho_mat.
sr_composites Optional vector selection ratios for composite variables. If not NULL, sr_composites must have as many elements as there are columns in wt_mat.
group_names Optional vector of group names.
var_names Optional vector of variable names.
composite_names Optional vector of names for composite variables.
diffs_as_obs Logical scalar that determines whether standard deviation parameters represent standard deviations of observed scores (TRUE) or of true scores (FALSE; default).

Value
A sample of simulated mean differences.

Examples

### Simulate statistics by providing integers as "n_vec":
simulate_d_sample(n_vec = c(200, 100), rho_mat_list = list(reshape_vec2mat(.5), reshape_vec2mat(.4)),
mu_mat = rbind(c(1, .5), c(0, 0)), sigma_mat = rbind(c(1, 1), c(1, 1)),
rel_mat = rbind(c(.8, .7), c(.7, .7)), sr_vec = c(1, .5),
group_names = c("A", "B"))

### Simulate parameters by providing proportions as "n_vec":
simulate_d_sample(n_vec = c(2/3, 1/3), rho_mat_list = list(reshape_vec2mat(.5), reshape_vec2mat(.4)),
mu_mat = rbind(c(1, .5), c(0, 0)), sigma_mat = rbind(c(1, 1), c(1, 1)),
rel_mat = rbind(c(.8, .7), c(.7, .7)), sr_vec = c(1, .5),
group_names = c("A", "B"))

---

simulate_matrix Generate a list of simulated sample matrices sampled from the Wishart distribution

---
**Description**

This function generates simulated sample matrices based on a population matrix and a sample size. It uses the Wishart distribution (i.e., the multivariate $\chi^2$ distribution) to obtain data, rescales the data into the input metric, and can be standardized into a correlation matrix by setting `as_cor` to `TRUE`. The function can produce a list of matrices for any number of samples.

**Usage**

```r
simulate_matrix(sigma, n, k = 1, as_cor = FALSE)
```

**Arguments**

- `sigma`: Population covariance matrix. May be standardized or unstandardized.
- `n`: Sample size for simulated sample matrices.
- `k`: Number of sample matrices to generate.
- `as_cor`: Should the simulated matrices be standardized (`TRUE`) or unstandardized (`FALSE`)?

**Value**

A list of simulated sample matrices.

**Examples**

```r
## Define a hypothetical matrix:
sigma <- reshape_vec2mat(cov = .4, order = 5)

## Simulate a list of unstandardized covariance matrices:
simulate_matrix(sigma = sigma, n = 50, k = 10, as_cor = FALSE)

## Simulate a list of correlation matrices:
simulate_matrix(sigma = sigma, n = 50, k = 10, as_cor = TRUE)
```

---

**simulate_psych**

*Simulate Monte Carlo psychometric data (observed, true, and error scores)*

**Description**

Simulate Monte Carlo psychometric data (observed, true, and error scores)

**Usage**

```r
simulate_psych(n, rho_mat, mu_vec = rep(0, ncol(rho_mat)),
               sigma_vec = rep(1, ncol(rho_mat)),
               sigma_factor = 1.0)
```
rel_vec = rep(1, ncol(rho_mat)),
sr_vec = rep(1, ncol(rho_mat)),
k_items_vec = rep(1, ncol(rho_mat)),
wt_mat = NULL,
sr_composites = NULL,
var_names = NULL,
composite_names = NULL
)

Arguments

n Number of cases to simulate before performing selection.
rho_mat Matrix of true-score correlations.
mu_vec Vector of means.
sigma_vec Vector of observed-score standard deviations.
rel_vec Vector of reliabilities corresponding to the variables in rho_mat.
sr_vec Vector of selection ratios corresponding to the variables in rho_mat. (set selection ratios to 1 for variables that should not be used in selection).
k_items_vec Number of test items comprising each of the variables to be simulated (all are single-item variables by default).
wt_mat Optional matrix of weights to use in forming a composite of the variables in rho_mat. Matrix should have as many rows (or vector elements) as there are variables in rho_mat.
sr_composites Optional vector selection ratios for composite variables. If not NULL, sr_composites must have as many elements as there are columns in wt_mat.
var_names Vector of variable names corresponding to the variables in rho_mat.
composite_names Optional vector of names for composite variables.

Value

A list of observed-score, true-score, and error-score data frames. If selection is requested, the data frames will include logical variables indicating whether each case would be selected on the basis of observed scores, true scores, or error scores.

Examples

## Generate data for a simple sample with two variables without selection:
simulate_psych(n = 1000, rho_mat = matrix(c(1, .5, .5, 1), 2, 2), sigma_vec = c(1, 1),
rel_vec = c(.8, .8), var_names = c("Y", "X"))

## Generate data for a simple sample with two variables with selection:
simulate_psych(n = 1000, rho_mat = matrix(c(1, .5, .5, 1), 2, 2), sigma_vec = c(1, 1),
rel_vec = c(.8, .8), sr_vec = c(1, .5), var_names = c("Y", "X"))

## Generate data for samples with five variables, of which subsets are used to form composites:
rho_mat <- matrix(.5, 5, 5)
The `simulate_r_database` function generates databases of psychometric correlation data from sample-size parameters, correlation parameters, reliability parameters, and selection-ratio parameters. The output database can be provided in either a long format or a wide format. If composite variables are to be formed, parameters can also be defined for the weights used to form the composites as well as the selection ratios applied to the composites. This function will return a database of statistics as well as a database of parameters - the parameter database contains the actual study parameters for each simulated sample (without sampling error) to allow comparisons between meta-analytic results computed from the statistics and the actual means and variances of parameters. The `merge_simdat_r` function can be used to merge multiple simulated databases and the `sparsify_simdat_r` function can be used to randomly delete artifact information (a procedure commonly done in simulations of artifact-distribution methods).

### Usage

```r
simulate_r_database(  
k,  
n_params,  
rho_params,  
mu_params = 0,  
sigma_params = 1,  
rel_params = 1,  
sr_params = 1,  
k_items_params = 1,  
wt_params = NULL,  
allow_neg_wt = FALSE,  
sr_composite_params = NULL,  
var_names = NULL,  
composite_names = NULL,  
n_as_ni = FALSE,  
show_applicant = FALSE,  
```

### Description

Simulate correlation databases of primary studies.
keep_vars = NULL,
decimals = 2,
format = "long",
max_iter = 100,
...  
)

Arguments

k           Number of studies to simulate.
n_params    Parameter distribution (or data-generation function; see details) for sample size.
rho_params  List of parameter distributions (or data-generation functions; see details) for correlations. If simulating data from a single fixed population matrix, that matrix can be supplied for this argument (if the diagonal contains non-unity values and 'sigma_params' is not specified, those values will be used as variances).
mu_params   List of parameter distributions (or data-generation functions; see details) for means.
sigma_params List of parameter distributions (or data-generation functions; see details) for standard deviations.
rel_params  List of parameter distributions (or data-generation functions; see details) for reliabilities.
sr_params   List of parameter distributions (or data-generation functions; see details) for selection ratios.
k_items_params List of parameter distributions (or data-generation functions; see details) for the number of test items comprising each of the variables to be simulated (all are single-item variables by default).
wt_params   List of parameter distributions (or data-generation functions; see details) to create weights for use in forming composites. If multiple composites are formed, the list should be a list of lists, with the general format: list(comp1_params = list(...params...), comp2_params = list(...params...), etc.).
allow_neg_wt Logical scalar that determines whether negative weights should be allowed (TRUE) or not (FALSE).
sr_composite_params Parameter distributions (or data-generation functions; see details) for composite selection ratios.
var_names   Optional vector of variable names for all non-composite variables.
composite_names Optional vector of names for composite variables.
n_as_ni     Logical argument determining whether n specifies the incumbent sample size (TRUE) or the applicant sample size (FALSE; default). This can only be TRUE when only one variable is involved in selection.
show_applicant Should applicant data be shown for sample statistics (TRUE) or suppressed (FALSE)?
keep_vars   Optional vector of variable names to be extracted from the simulation and returned in the output object. All variables are returned by default. Use this argument when only some variables are of interest and others are generated solely to serve as selection variables.
decimals Number of decimals to which statistical results (not parameters) should be rounded. Rounding to 2 decimal places best captures the precision of data available from published primary research.

format Database format: "long" or "wide."

max_iter Maximum number of iterations to allow in the parameter selection process before terminating with convergence failure. Must be finite.

... Additional arguments.

Details

Values supplied as any argument with the suffix "params" can take any of three forms (see Examples for a demonstration of usage):

- A vector of values from which study parameters should be sampled.
- A vector containing a mean with a variance or standard deviation. These values must be named "mean," "var," and "sd," respectively, for the program to recognize which value is which.
- A matrix containing a row of values (this row must be named "values") from which study parameters should be sampled and a row of weights (this row must be labeled 'weights') associated with the values to be sampled.
- A matrix containing a column of values (this column must be named "values") from which study parameters should be sampled and a column of weights (this column must be labeled 'weights') associated with the values to be sampled.
- A function that is configured to generate data using only one argument that defines the number of cases to generate, e.g., fun(n = 10).

Value

A database of simulated primary studies' statistics and analytically determined parameter values.

Examples

## Not run:
## Note the varying methods for defining parameters:
geomean = function(n) rgamma(n, shape = 100)
rho_params <- list(c(.1, .3, .5),
                  c(mean = .3, sd = .05),
                  rbind(value = c(.1, .3, .5), weight = c(1, 2, 1)))
rel_params = list(c(.7, .8, .9),
                  c(mean = .8, sd = .05),
                  rbind(value = c(.7, .8, .9), weight = c(1, 2, 1)))
sr_params = list(c(1, 1, c(.5, .7)))
sr_composite_params = list(c(1, 1, c(.5, .6, .7)))
wt_params = list(c(1, 2, 3),
                 c(mean = 2, sd = .25),
                 rbind(value = c(1, 2, 3), weight = c(1, 2, 1))),
                list(c(1, 2, 3),
                     c(mean = 2, sd = .25),
                     cbind(value = c(1, 2, 3), weight = c(1, 2, 1))))
## Simulate with long format

```r
simulate_r_database(k = 10, n_params = n_params, rho_params = rho_params,
                    rel_params = rel_params, sr_params = sr_params,
                    sr_composite_params = sr_composite_params, wt_params = wt_params,
                    var_names = c("X", "Y", "Z"), format = "long")
```

## Simulate with wide format

```r
simulate_r_database(k = 10, n_params = n_params, rho_params = rho_params,
                    rel_params = rel_params, sr_params = sr_params,
                    sr_composite_params = sr_composite_params, wt_params = wt_params,
                    var_names = c("X", "Y", "Z"), format = "wide")
```

## End(Not run)

---

**simulate_r_sample**  
*Simulation of data with measurement error and range-restriction artifacts*

### Description

This function simulates a psychometric sample and produces correlation matrices, artifact information, and other descriptive statistics that have been affected by measurement error and/or range restriction. It allows the formation of composite variables within the simulation and allows selection to be performed on any or all variables, including composites. By setting the sample size to \( n = \infty \), users can explore the effects of measurement error and/or range restriction on parameters without the influence of sampling error. To generate multiple samples and compile a database of simulated statistics, see the `simulate_r_database` function.

### Usage

```r
simulate_r_sample(
  n,
  rho_mat,
  rel_vec = rep(1, ncol(rho_mat)),
  mu_vec = rep(0, ncol(rho_mat)),
  sigma_vec = rep(1, ncol(rho_mat)),
  sr_vec = rep(1, ncol(rho_mat)),
  k_items_vec = rep(1, ncol(rho_mat)),
  wt_mat = NULL,
  sr_composites = NULL,
  var_names = NULL,
  composite_names = NULL,
  n_as_ni = FALSE,
  ...
)
```
simulate_r_sample

Arguments

- **n**: Number of cases to simulate before performing selection. If Inf, function will simulate parameter values.
- **rho_mat**: Matrix of true-score correlations.
- **rel_vec**: Vector of reliabilities corresponding to the variables in rho_mat.
- **mu_vec**: Vector of means.
- **sigma_vec**: Vector of observed-score standard deviations.
- **sr_vec**: Vector of selection ratios corresponding to the variables in rho_mat (set selection ratios to 1 for variables that should not be used in selection).
- **k_items_vec**: Number of test items comprising each of the variables to be simulated (all are single-item variables by default).
- **wt_mat**: Optional matrix of weights to use in forming a composite of the variables in rho_mat. Matrix should have as many rows (or vector elements) as there are variables in rho_mat.
- **sr_composites**: Optional vector selection ratios for composite variables. If not NULL, sr_composites must have as many elements as there are columns in wt_mat.
- **var_names**: Vector of variable names corresponding to the variables in rho_mat.
- **composite_names**: Optional vector of names for composite variables.
- **n_as_ni**: Logical argument determining whether n specifies the incumbent sample size (TRUE) or the applicant sample size (FALSE; default). This can only be TRUE when only one variable is involved in selection.
- **...**: Further arguments.

Value

A list of study information, including correlations, reliabilities, standard deviations, means, and u ratios for true scores and for observed scores.

Examples

```r
## Not run:
## Generate data for a simple sample with two variables:
simulate_r_sample(n = 1000, rho_mat = matrix(c(1, .5, .5, 1), 2, 2),
                   rel_vec = c(.8, .8), sr_vec = c(1, .5), var_names = c("Y", "X"))

## Generate data for samples with five variables, of which subsets are used to form composites:
rho_mat <- matrix(.5, 5, 5)
diag(rho_mat) <- 1

## Simulate parameters by supply n = Inf
simulate_r_sample(n = Inf, rho_mat = rho_mat,
                   rel_vec = rep(.8, 5), sr_vec = c(1, 1, 1, 1, .5),
                   wt_mat = cbind(c(0, 0, 0, .3, 1), c(1, .3, 0, 0, 0)), sr_composites = c(.7, .5))

## Finite sample sizes allow the generation of sample data
```
sparsify_simdat_d

Create sparse artifact information in a "simdat_d_database" class object

Description

This function can be used to randomly delete artifact from databases produced by the `simulate_d_database` function. Deletion of artifacts can be performed in either a study-wise fashion for complete missingness within randomly selected studies or element-wise missingness for completely random deletion of artifacts in the database. Deletion can be applied to reliability estimates and/or u ratios.

Usage

```r
sparsify_simdat_d(
  data_obj,
  prop_missing,
  sparify_arts = c("rel", "u"),
  study_wise = TRUE
)
```

Arguments

- `data_obj`: Object created by the "simdat_d_database" function.
- `prop_missing`: Proportion of studies in from which artifact information should be deleted.
- `sparify_arts`: Vector of codes for the artifacts to be sparsified: "rel" for reliabilities, "u" for u ratios, or c("rel", "u") for both.
- `study_wise`: Logical scalar argument determining whether artifact deletion should occur for all variables in a study (TRUE) or randomly across variables within studies (FALSE).

Value

A sparsified database
sparsify_simdat_r  

Create sparse artifact information in a "simdat_r_database" class object

Description

This function can be used to randomly delete artifact from databases produced by the simulate_r_database function. Deletion of artifacts can be performed in either a study-wise fashion for complete missingness within randomly selected studies or element-wise missingness for completely random deletion of artifacts in the database. Deletion can be applied to reliability estimates and/or u ratios.

Usage

sparsify_simdat_r(
  data_obj,
  prop_missing,
  sparify_arts = c("rel", "u"),
  study_wise = TRUE
)

Arguments

data_obj Object created by the "simdat_r_database" function.
prop_missing Proportion of studies in from which artifact information should be deleted.
sparify_arts Vector of codes for the artifacts to be sparsified: "rel" for reliabilities, "u" for u ratios, or c("rel", "u") for both.
study_wise Logical scalar argument determining whether artifact deletion should occur for all variables in a study (TRUE) or randomly across variables within studies (FALSE).

Value

A sparsified database

summary  

Summary methods for psychmeta

Description

Summary methods for psychmeta output objects with classes exported from psychmeta.

Arguments

object Object to be printed (object is used to select a method).
... Additional arguments.
Value

Summary object.

| truncate_dist | Truncation function for normal distributions (truncates both mean and variance) |

Description

This function computes the mean and variance of a normal distributions that has been truncated at one or both ends.

Usage

```r
truncate_dist(a = -Inf, b = Inf, mean = 0, sd = 1)
```

Arguments

- `a`: Quantile (i.e., cut score) below which scores should be censored from the distribution.
- `b`: Quantile (i.e., cut score) above which scores should be censored from the distribution.
- `mean`: Scalar mean or vector of means.
- `sd`: Scalar standard deviation or vector of standard deviations.

Value

A matrix of truncated means (column 1) and truncated variances (column 2).

Examples

```r
truncate_dist(a = -1, b = 3, mean = 0, sd = 1)
truncate_dist(a = 1, b = Inf, mean = 0, sd = 1)
truncate_dist(a = c(-1, 1), b = c(3, Inf), mean = 0, sd = 1)
```
**truncate_mean**  
*Truncation function for means*

**Description**

This function computes the mean of a normal distribution that has been truncated at one or both ends.

**Usage**

```r
truncate_mean(a = -Inf, b = Inf, mean = 0, sd = 1)
```

**Arguments**

- `a`: Quantile (i.e., cut score) below which scores should be censored from the distribution.
- `b`: Quantile (i.e., cut score) above which scores should be censored from the distribution.
- `mean`: Scalar mean or vector of means.
- `sd`: Scalar standard deviation or vector of standard deviations.

**Value**

A vector of truncated means.

**Examples**

```r
truncate_mean(a = -1, b = 3, mean = 0, sd = 1)
truncate_mean(a = 1, b = Inf, mean = 0, sd = 1)
truncate_mean(a = c(-1, 1), b = c(3, Inf), mean = 0, sd = 1)
```

---

**truncate_var**  
*Truncation function for variances*

**Description**

This function computes the variance of a normal distribution that has been truncated at one or both ends.

**Usage**

```r
truncate_var(a = -Inf, b = Inf, mean = 0, sd = 1)
```
Arguments

a  Quantile (i.e., cut score) below which scores should be censored from the distribution.
b  Quantile (i.e., cut score) above which scores should be censored from the distribution.
mean  Scalar mean or vector of means.
sd  Scalar standard deviation or vector of standard deviations.

Value
A vector of truncated variances

Examples

```r
truncate_var(a = -1, b = 3, mean = 0, sd = 1)
truncate_var(a = 1, b = Inf, mean = 0, sd = 1)
truncate_var(a = c(-1, 1), b = c(3, Inf), mean = 0, sd = 1)
```

---

unmix_matrix  Estimate average within-group covariance matrices from a mixture covariance matrix

Description
Estimate average within-group covariance matrices from a mixture covariance matrix

Usage

```r
unmix_matrix(
  sigma_mat,
  mu_mat,
  p_vec,
  N = Inf,
  group_names = NULL,
  var_names = NULL
)
```

Arguments

sigma_mat  Mixture covariance matrix.
mu_mat  Matrix of mean parameters, with groups on the rows and variables on the columns.
p_vec  Vector of proportion of cases in each group.
N  Optional total sample size across all groups (used to compute unbiased covariance estimates).
group_names  Optional vector of group names.
var_names  Optional vector of variable names.
unmix_r_2group

Value

List of within-group covariances and means.

Examples

```r
out <- unmix_matrix(sigma_mat = reshape_vec2mat(.5, order = 2),
                      mu_mat = rbind(c(0, 0), c(.5, 1)),
                      p_vec = c(.3, .7), N = 100)
```

## Result of unmix_matrix:
out

## Simulated data reproduce the total parameter matrix:
dat <- NULL
for(i in 1:2){
  dat <- rbind(dat, cbind(group = i,
                           data.frame(MASS::mvrnorm(n = round(out$p_group[i] * out$N),
                                                   mu = out$means_raw[i,],
                                                   Sigma = out$cov_group_unbiased[[i]],
                                                   empirical = TRUE))))
}
cov(dat[, -1])
```

unmix_r_2group  Estimate the average within-group correlation from a mixture correlation for two groups

Description

Estimate the average within-group correlation from a mixture correlation for two groups.

Usage

```r
unmix_r_2group(rxy, dx, dy, p = 0.5)
```

Arguments

- `rxy` Overall mixture correlation.
- `dx` Standardized mean difference between groups on X.
- `dy` Standardized mean difference between groups on Y.
- `p` Proportion of cases in one of the two groups.
Details

The mixture correlation for two groups is estimated as:

\[ r_{xy_{Mix}} = \rho_{xy_{WG}} + \sqrt{d_x^2 d_y^2 p^2 (1 - p)^2} \]

\[ \frac{1}{\sqrt{(d_x^2 p(1 - p) + 1)(d_y^2 p(1 - p) + 1)}} \]

where \( \rho_{xy_{WG}} \) is the average within-group correlation, \( \rho_{xy_{Mix}} \) is the overall mixture correlation, \( d_x \) is the standardized mean difference between groups on X, \( d_y \) is the standardized mean difference between groups on Y, and \( p \) is the proportion of cases in one of the two groups.

Value

A vector of average within-group correlations

References


Examples

unmix_r_2group(rxy = .5, dx = 1, dy = 1, p = .5)

<table>
<thead>
<tr>
<th>var_error_A</th>
<th>Estimate the error variance of the probability-based effect size (A, AUC, the common language effect size [CLES])</th>
</tr>
</thead>
</table>

Description

Estimates the error variance of the probability-based common language effect size (A, AUC, CLES)

Usage

var_error_A(A, n1, n2 = NA)

var_error_auc(A, n1, n2 = NA)

var_error_cles(A, n1, n2 = NA)

Arguments

A     Vector of probability-based effect sizes (common language effect sizes)

n1    Vector of sample sizes from group 1 (or the total sample size with the assumption that groups are of equal size, if no group 2 sample size is supplied).

n2    Vector of sample sizes from group 2.
Details

The sampling variance of a \( A \) (also called \( AUC \) [area under curve] or \( CLES \) [common-language effect size]) value is:

\[
\frac{\left( \frac{1}{n_1} \right) + \left( \frac{1}{n_2} \right) + \left( \frac{1}{n_1 n_2} \right)}{12}
\]

When groups 1 and 2 are of equal size, this reduces to

\[
\frac{\left( \frac{1}{n} \right) + \left( \frac{1}{n^2} \right)}{3}
\]

Value

A vector of sampling-error variances.

References


Examples

```r
var_error_A(A = 1, n1 = 30, n2 = 30)
var_error_auc(A = 1, n1 = 60, n2 = NA)
var_error_cles(A = 1, n1 = 30, n2 = 30)
```

---

**var_error_alpha**  
**Analytic estimate of the sampling variance of coefficient \( \alpha \)**

Description

Estimates the error variance of Cronbach’s coefficient \( \alpha \).

Usage

```r
var_error_alpha(item_mat = NULL, alpha = NULL, k_items = NULL, n_cases)
```

Arguments

- **item_mat**: Item correlation/covariance matrix. If item_mat is not supplied, the user must supply both alpha and k_items. If item_mat is NULL, the program will assume that all item intercorrelations are equal.
- **alpha**: Vector of population \( \alpha \) values. Must be supplied if item_mat is NULL.
- **k_items**: Vector of numbers of items to be simulated. Must be supplied if item_mat is NULL.
- **n_cases**: Vector of sample sizes to simulate in sampling distribution of alpha.
**Value**

Vector of sampling variances of the supplied $\alpha$ values.

**References**


**Examples**

```r
item_mat <- matrix(.3, 5, 5)
diag(item_mat) <- 1
alpha <- mean(item_mat[lower.tri(item_mat)]) / mean(item_mat)
k_items <- nrow(item_mat)

var_error_alpha(item_mat = item_mat, n_cases = 50)
var_error_alpha(alpha = alpha, k_items = k_items, n_cases = 50)
var_error_alpha(alpha = c(alpha, alpha), k_items = c(k_items, k_items), n_cases = 50)
```

---

**var_error_d**

Estimate the error variance Cohen’s $d$ values

**Description**

Estimates the error variance of standardized mean differences (Cohen’s $d$ values)

**Usage**

```r
var_error_d(d, n1, n2 = NA, correct_bias = TRUE)
```

**Arguments**

- **d**: Vector of Cohen’s $d$ values.
- **n1**: Vector of sample sizes from group 1 (or the total sample size with the assumption that groups are of equal size, if no group 2 sample size is supplied).
- **n2**: Vector of sample sizes from group 2.
- **correct_bias**: Logical argument that determines whether to correct error-variance estimates for small-sample bias in $d$ values (TRUE) or not (FALSE).

**Details**

Allows for error variance to be estimated using total sample size of both groups being compared (in this case, supply sample sizes using only the n1 argument) or using separate sample sizes for group 1 and group 2 (i.e., the groups being compared; in this case, supply sample sizes using both the n1 and n2 arguments).

The sampling variance of a $d$ value is:
\[
\frac{(n - 1)}{(n - 3)} \left( \frac{n_1 + n_2}{n_1 n_2} + \frac{d^2}{2(n_1 + n_2)} \right)
\]

When groups 1 and 2 are of equal size, this reduces to

\[
\text{var}_e = \left( \frac{n - 1}{n - 3} \right) \left( \frac{4}{n} \right) \left( 1 + \frac{d^2}{8} \right)
\]

This can be corrected for bias by first correcting the \(d\) value (see \texttt{correct_d_bias()}) prior to estimating the error variance.

**Value**

A vector of sampling-error variances.

**References**


**Examples**

```R
var_error_d(d = 1, n1 = 30, n2 = 30, correct_bias = TRUE)
var_error_d(d = 1, n1 = 60, n2 = NA, correct_bias = TRUE)
```

---

**Description**

Estimates the error variance of standardized mean differences (Glass’s \(\Delta\) values)

**Usage**

```R
var_error_delta(d, nc, ne = NA, use_pooled_sd = FALSE, correct_bias = TRUE)
```

**Arguments**

- `delta` Vector of Glass’ \(\Delta\) values.
- `nc` Vector of control-group sample sizes (or the total sample size with the assumption that groups are of equal size, if no experimental-group sample size is supplied).
- `ne` Vector of experimental-group sample sizes.
- `use_pooled_sd` Logical vector determining whether the pooled standard deviation was used (‘TRUE’) or not (‘FALSE’, default).
- `correct_bias` Logical argument that determines whether to correct error-variance estimates for small-sample bias in \(d\) values (‘TRUE’) or not (‘FALSE’).
A vector of sampling-error variances.

Examples

```r
var_error_delta(delta = .3, nc = 30, ne = 30)
var_error_delta(delta = .3, nc = 60, ne = NA)
```

---

### Description

Allows for error variance to be estimated using total sample size of both groups being compared (in this case, supply sample sizes using only the n1 argument) or using separate sample sizes for group 1 and group 2 (i.e., the groups being compared; in this case, supply sample sizes using both the n1 and n2 arguments).

### Usage

```r
var_error_g(g, n1, n2 = NA, a_method = c("gamma", "approx"))
```

### Arguments

- **g**: Vector of Hedges’s g values.
- **n1**: Vector of sample sizes from group 1 (or the total sample size with the assumption that groups are of equal size, if no group 2 sample size is supplied).
- **n2**: Vector of sample sizes from group 2.
- **a_method**: Method used to correct the bias in Cohen’s d to convert to Hedges’s g. Options are “gamma” (default) for the exact method based on the gamma function (Hedges & Olkin, 1985) or “approx” for the computationally trivial approximation (Borenstein et al., 2006).

### Value

A vector of sampling-error variances.

### References


### Examples

```r
var_error_g(g = 1, n1 = 30, n2 = 30)
var_error_g(g = 1, n1 = 60, n2 = NA)
```
Estimate the error variance of linear regression multiple $R$(-squared)

**Description**

This function estimates the error variance for linear regression model (squared) multiple correlations ($R$ and $R^2$).

**Usage**

- `var_error_mult_R(R, n, p)`
- `var_error_mult_Rsq(Rsq, n, p)`
- `var_error_R(R, n, p)`
- `var_error_Rsq(Rsq, n, p)`

**Arguments**

- `R` Vector of multiple correlation coefficients.
- `n` Vector of sample sizes.
- `p` Vector of numbers of predictors in the model.
- `Rsq` Vector of squared multiple correlation coefficients.

**Details**

The sampling variance of a multiple correlation is approximately:

$$\text{var}_e = \frac{(1 - R^2)^2(n - p - 1)^2}{(n^2 - 1)(n + 3)}$$

The sampling variance of a squared multiple correlation is approximately:

$$\text{var}_e = \frac{4R^2(1 - R^2)^2(n - p - 1)^2}{(n^2 - 1)(n + 3)}$$

**Value**

A vector of sampling-error variances.

**References**

**Examples**

```r
case_error_mult_R(R = .5, n = 30, p = 4)
case_error_mult_R(R = .5, n = 30, p = 4)
case_error_mult_Rsq(Rsq = .25, n = 30, p = 4)
case_error_mult_Rsq(Rsq = .25, n = 30, p = 4)
```

---

**var_error_q**  
*Estimate the error variance of square roots of reliability estimates*

**Description**

Estimate error variance for square-root reliability coefficients (measure quality indices; \(\sqrt{\rho_{xx}}\) or \(q_{Xx}\)).

**Usage**

```r
var_error_q(q, n, rel_type = "alpha", k_items = NULL)
```

**Arguments**

- `q`  
  Vector of square roots of reliability estimates.

- `n`  
  Vector of sample sizes.

- `rel_type`  
  Character vector indicating the type(s) of reliabilities being analyzed. See documentation for `ma_r()` for a full list of acceptable reliability types. NOTE: Currently, only alpha has its own dedicated error-variance estimate; the error variance of other reliability types is estimated using the generic definition of reliability as the squared correlation between observed scores and true scores.

- `k_items`  
  Optional numeric vector indicating the number of items in each scale for which reliabilities are being analyzed.

**Details**

The sampling variance of the square root of a reliability coefficient is:

\[
\text{var}_e = \frac{(1 - q_{Xx}^2)^2}{n - 1}
\]

For the equation to estimate the variance of coefficient alpha, see Duhachek and Iacobucci (2004).

**Value**

A vector of sampling-error variances.
References


Examples

```r
var_error_q(q = .8, n = 100)
var_error_q(q = .8, n = 100, rel_type = "alpha", k_items = 10)
```

---

**var_error_r**

Estimate the error variance of correlations

### Description

Estimates the error variance of Pearson correlations ($r$).

### Usage

```r
var_error_r(r, n, correct_bias = TRUE)
```

### Arguments

- **r**
  - Vector of correlations.
- **n**
  - Vector of sample sizes.
- **correct_bias**
  - Logical argument that determines whether to correct error-variance estimates for small-sample bias in correlations (TRUE) or not (FALSE).

### Details

The sampling variance of a Pearson correlation is approximately:

$$\text{var}_r = \frac{(1 - r^2)^2}{n - 1}$$

This can be corrected for bias in the sample correlation by first correcting the correlation (see `correct_r_bias()`) prior to estimating the error variance.

### Value

A vector of sampling-error variances.
References


Examples

```r
var_error_r(r = .3, n = 30, correct_bias = TRUE)
var_error_r(r = .3, n = 30, correct_bias = FALSE)
```

---

**var_error_rel**

*Estimate the error variance of reliability estimates*

**Description**

Estimate error variance for reliability coefficients ($r_{XX}$).

**Usage**

```r
var_error_rel(rel, n, rel_type = "alpha", k_items = NULL)
```

**Arguments**

- `rel` Vector of reliability estimates.
- `n` Vector of sample sizes.
- `rel_type` Character vector indicating the type(s) of reliabilities being analyzed. See documentation for `ma_r()` for a full list of acceptable reliability types. NOTE: Currently, only $\alpha$ has its own dedicated error-variance estimate; the error variance of other reliability types is estimated using the generic definition of reliability as the squared correlation between observed scores and true scores.
- `k_items` Optional numeric vector indicating the number of items in each scale for which reliabilities are being analyzed.

**Details**

The sampling variance of a reliability coefficient is:

$$\text{var}_e = \frac{4r_{XX}(1 - r_{XX})^2}{n - 1}$$

For the equation to estimate the variance of coefficient $\alpha$, see Duhachek and Iacobucci (2004).

**Value**

A vector of sampling-error variances.
References


Examples

```r
var_error_rel(rel = .8, n = 100)
var_error_rel(rel = .8, n = 100, rel_type = "alpha", k_items = 10)
```

Description

This function propagates error in the bivariate indirect range-restriction correction formula to allow for the computation of a pseudo compound attenuation factor in individual-corrrection meta-analysis. Traditional methods for estimating compound attenuation factors (i.e., dividing the observed correlation by the corrected correlation) do not work with the BVIRR correction because BVIRR has an additive term that makes the corrected correlation inappropriate for use in estimating the effect of the correction on the variance of the sampling distribution of correlations. The equation-implied adjustment for the BVIRR correction (i.e., the first derivative of the correction equation with respect to the observed correlation) underestimates the error of corrected correlations, so this function helps to account for that additional error.

Usage

```r
var_error_r_bvirr(
  rxyi,
  var_e = NULL,
  ni,
  na = NA,
  ux = rep(1, length(rxyi)),
  ux_observed = rep(TRUE, length(rxyi)),
  uy = rep(1, length(rxyi)),
  uy_observed = rep(TRUE, length(rxyi)),
  qx = rep(1, length(rxyi)),
  qx_restricted = rep(TRUE, length(rxyi)),
  qx_type = rep("alpha", length(rxyi)),
  k_items_x = rep(NA, length(rxyi)),
  qy = rep(1, length(rxyi)),
)```
```r
qy_restricted = rep(TRUE, length(rxyi)),
qy_type = rep("alpha", length(rxyi)),
k_items_y = rep(NA, length(rxyi)),
mean_rxyi = NULL,
mean_ux = NULL,
mean_uy = NULL,
mean_qxa = NULL,
mean_qya = NULL,
var_rxyi = NULL,
var_ux = NULL,
var_uy = NULL,
var_qxa = NULL,
var_qya = NULL,
cor_rxyi_ux = 0,
cor_rxyi_uy = 0,
cor_rxyi_qxa = 0,
cor_rxyi_qya = 0,
cor_ux_uy = 0,
cor_ux_qxa = 0,
cor_ux_qya = 0,
cor_uy_qxa = 0,
cor_uy_qya = 0,
cor_qxa_qya = 0,
sign_rxz = 1,
sign_ryz = 1,
r_deriv_only = FALSE
```

**Arguments**

- **rxyi**: Vector of observed correlations.
- **var_e**: Vector of estimated sampling variances for rxyi values.
- **ni**: Vector of incumbent sample sizes (necessary when variances of correlations/artifacts are not supplied).
- **na**: Optional vector of applicant sample sizes (for estimating error variance of u ratios and applicant reliabilities).
- **ux**: Vector of observed-score u ratios for X.
- **ux_observed**: Logical vector in which each entry specifies whether the corresponding ux value is an observed-score u ratio (TRUE) or a true-score u ratio. All entries are TRUE by default.
- **uy**: Vector of observed-score u ratios for Y.
- **uy_observed**: Logical vector in which each entry specifies whether the corresponding uy value is an observed-score u ratio (TRUE) or a true-score u ratio. All entries are TRUE by default.
- **qx**: Vector of square roots of reliability estimates for X.
- **qx_restricted**: Logical vector determining whether each element of qx is derived from an incumbent reliability (TRUE) or an applicant reliability (FALSE).
var_error_r_bvrr

qx_type, qy_type
String vector identifying the types of reliability estimates supplied (e.g., "alpha", "retest", "interrater_r", "splithalf"). See the documentation for `ma_r()` for a full list of acceptable reliability types.

k_items_x, k_items_y
Numeric vector identifying the number of items in each scale.

qy
Vector of square roots of reliability estimates for X.

qy_restricted
Logical vector determining whether each element of qy is derived from an incumbent reliability (TRUE) or an applicant reliability (FALSE).

mean_rxyi
Mean observed correlation.

mean_ux
Mean observed-score u ratio for X (for use in estimating sampling errors in the context of a meta-analysis).

mean_uy
Mean observed-score u ratio for Y (for use in estimating sampling errors in the context of a meta-analysis).

mean_qxa
Mean square-root applicant reliability estimate for X (for use in estimating sampling errors in the context of a meta-analysis).

mean_qya
Mean square-root applicant reliability estimate for Y (for use in estimating sampling errors in the context of a meta-analysis).

var_rxyi
Optional pre-specified variance of correlations.

var_ux
Optional pre-specified variance of observed-score u ratios for X.

var_uy
Optional pre-specified variance of observed-score u ratios for Y.

var_qxa
Optional pre-specified variance of square-root applicant reliability estimate for X.

var_qya
Optional pre-specified variance of square-root applicant reliability estimate for Y.

cor_rxyi_ux
Correlation between rxyi and ux (zero by default).

cor_rxyi_uy
Correlation between rxyi and uy (zero by default).

cor_rxyi_qxa
Correlation between rxyi and qxa (zero by default).

cor_rxyi_qya
Correlation between rxyi and qya (zero by default).

cor_ux_uy
Correlation between ux and uy (zero by default).

cor_ux_qxa
Correlation between ux and qxa (zero by default).

cor_ux_qya
Correlation between ux and qya (zero by default).

cor_uy_qxa
Correlation between uy and qxa (zero by default).

cor_uy_qya
Correlation between uy and qya (zero by default).

cor_qxa_qya
Correlation between qxa and qya (zero by default).

sign_rxz
Sign of the relationship between X and the selection mechanism.

sign_ryz
Sign of the relationship between Y and the selection mechanism.

r_deriv_only
Logical scalar determining whether to use the partial derivative with respect to rxyi only (TRUE) or a full Taylor series approximation of the disattenuation formula (FALSE).
Details

Per the principles of propagation of uncertainty and assuming that \( q_x, q_y, u_x, u_y, \) and \( \rho_{XY} \), are independent, we can derive a linear approximation of the sampling error of \( \rho_{TP_n} \). We begin with the bivariate indirect range restriction formula,

\[
\rho_{TP_n} = \frac{\rho_{XY} u_X u_Y + \lambda \sqrt{|1 - u_X^2| |1 - u_Y^2|}}{q_X q_Y}
\]

which implies the following linear approximation of the sampling variance of \( \rho_{TP_n} \):

\[
SE_{\rho_{TP_n}}^2 = b_1^2 SE_{\rho_{XY}}^2 + b_2^2 SE_{q_X}^2 + b_3^2 SE_{u_X}^2 + b_4^2 SE_{u_Y}^2 + b_5^2 SE_{\rho_{XY}}^2
\]

where \( b_1, b_2, b_3, b_4, \) and \( b_5 \) are the first-order partial derivatives of the disattenuation formula with respect to \( q_x, q_y, u_x, u_y, \) and \( \rho_{XY} \), respectively. These partial derivatives are computed as follows:

\[
b_1 = \frac{\partial \rho_{TP_n}}{\partial q_x} = -\frac{\rho_{TP_n} u_X}{q_X}
\]

\[
b_2 = \frac{\partial \rho_{TP_n}}{\partial q_y} = -\frac{\rho_{TP_n} u_Y}{q_Y}
\]

\[
b_3 = \frac{\partial \rho_{TP_n}}{\partial u_X} = \left[ \rho_{XY} u_Y - \frac{\lambda u_X (1 - u_Y^2) \sqrt{1 - u_X^2}}{|1 - u_X^2|^{1.5}} \right] / (q_X q_Y)
\]

\[
b_4 = \frac{\partial \rho_{TP_n}}{\partial u_Y} = \left[ \rho_{XY} u_X - \frac{\lambda u_Y (1 - u_X^2) \sqrt{1 - u_Y^2}}{|1 - u_Y^2|^{1.5}} \right] / (q_X q_Y)
\]

\[
b_5 = \frac{\partial \rho_{TP_n}}{\partial \rho_{XY}} = \frac{u_X u_Y}{q_X q_Y}
\]

Value

A vector of corrected correlations’ sampling-error variances.

References


Examples

```r
var_error_r_bvirr(rxyi = .3, var_e = var_error_r(r = .3, n = 100), ni = 100,
ux = .8, uy = .8,
qx = .9, qx_restricted = TRUE,
qy = .9, qy_restricted = TRUE,
sign_rxz = 1, sign_ryz = 1)
```
Estimate the error variance of Spearman rank correlations

Description

Estimates the variance of Spearman rank correlations ($\rho$) using the Fieller correction.

Usage

```r
var_error_spearman(r, n, correct_bias = TRUE)
```

Arguments

- `r`: Vector of rank correlations.
- `n`: Vector of sample sizes.
- `correct_bias`: Logical argument that determines whether to correct error-variance estimates for small-sample bias in correlations (TRUE) or not (FALSE).

Details

The sampling variance of a Spearman rank correlation is approximately:

$$ var_e = \frac{1.06 \times (1 - r^2)^2}{n - 1} $$

This can be corrected for bias in the sample correlation by first correcting the correlation (see `correct_r_bias()`) prior to estimating the error variance.

Value

A vector of sampling-error variances.

References


Examples

```r
var_error_spearman(r = .3, n = 30, correct_bias = TRUE)
var_error_spearman(r = .3, n = 30, correct_bias = FALSE)
```
var_error_u

Estimate the error variance of \( u \) ratios

Description

Estimates the error variance of standard deviation (\( u \)) ratios.

Usage

\[
\text{var_error_u}(u, \text{ni}, \text{na} = \text{NA}, \text{dependent_sds} = \text{FALSE})
\]

Arguments

\begin{itemize}
  \item \( u \): Vector of \( u \) ratios.
  \item \( \text{ni} \): Vector of incumbent-group sample sizes.
  \item \( \text{na} \): Vector of applicant-group sample sizes.
  \item \( \text{dependent_sds} \): Logical vector identifying whether each \( u \) ratio is based on standard deviations from independent samples (\text{FALSE}) or based on standard deviations from an applicant sample and an incumbent sample that is a subset of that applicant sample (\text{TRUE}).
\end{itemize}

Details

The sampling variance of a \( u \) ratio is computed differently for independent samples (i.e., settings where the referent unrestricted standard deviation comes from a different sample than the range-restricted standard deviation) than for dependent samples (i.e., unrestricted samples from which a subset of individuals are selected to be in the incumbent sample).

The sampling variance for independent samples (the more common case) is:

\[
\text{var}_e = \frac{u^2}{2} \left( \frac{1}{n_i - 1} + \frac{1}{n_a - 1} \right)
\]

and the sampling variance for dependent samples is:

\[
\text{var}_e = \frac{u^2}{2} \left( \frac{1}{n_i - 1} - \frac{1}{n_a - 1} \right)
\]

where \( u \) is the \( u \) ratio, \( n_i \) is the incumbent sample size, and \( n_a \) is the applicant sample size.

Value

A vector of sampling-error variances.

References

Examples

```r
var_error_u(u = .8, ni = 100, na = 200)
var_error_u(u = .8, ni = 100, na = NA)
```

### wt_cov

**wt_cov**

*Compute weighted covariances*

**Description**

Compute the weighted covariance among variables in a matrix or between the variables in two separate matrices/vectors.

**Usage**

```r
wt_cov(
  x,
  y = NULL,
  wt = NULL,
  as_cor = FALSE,
  use = c("everything", "listwise", "pairwise"),
  unbiased = TRUE
)
```

```r
wt_cor(x, y = NULL, wt = NULL, use = "everything")
```

**Arguments**

- `x` Vector or matrix of x variables.
- `y` Vector or matrix of y variables
- `wt` Vector of weights
- `as_cor` Logical scalar that determines whether the covariances should be standardized (TRUE) or unstandardized (FALSE).
- `use` Method for handling missing values. "everything" uses all values and does not account for missingness, "listwise" uses only complete cases, and "pairwise" uses pairwise deletion.
- `unbiased` Logical scalar determining whether variance should be unbiased (TRUE) or maximum-likelihood (FALSE).

**Value**

Scalar, vector, or matrix of covariances.
Examples

wt_cov(x = c(1, 0, 2), y = c(1, 2, 3), wt = c(1, 2, 2), as_cor = FALSE, use = "everything")
wt_cov(x = c(1, 0, 2), y = c(1, 2, 3), wt = c(1, 2, 2), as_cor = TRUE, use = "everything")
wt_cov(x = cbind(c(1, 0, 2), c(1, 2, 3)), wt = c(1, 2, 2), as_cor = FALSE, use = "everything")
wt_cov(x = cbind(c(1, 0, 2), c(1, 2, 3)), wt = c(1, 2, 2), as_cor = TRUE, use = "everything")
wt_cov(x = cbind(c(1, 0, 2, NA), c(1, 2, 3, 3)),
       wt = c(1, 2, 2, 1), as_cor = FALSE, use = "listwise")
wt_cov(x = cbind(c(1, 0, 2, NA), c(1, 2, 3, 3)),
       wt = c(1, 2, 2, 1), as_cor = TRUE, use = "listwise")

wt_dist

Weighted descriptive statistics for a vector of numbers

Description

Compute the weighted mean and variance of a vector of numeric values. If no weights are supplied, defaults to computing the unweighted mean and the unweighted maximum-likelihood variance.

Usage

wt_dist(
  x,
  wt = rep(1, length(x)),
  unbiased = TRUE,
  df_type = c("count", "sum_wts")
)

wt_mean(x, wt = rep(1, length(x)))

wt_var(
  x,
  wt = rep(1, length(x)),
  unbiased = TRUE,
  df_type = c("count", "sum_wts")
)

Arguments

x Vector of values to be analyzed.
wt Weights associated with the values in x.
unbiased Logical scalar determining whether variance should be unbiased (TRUE) or maximum-likelihood (FALSE).
df_type Character scalar determining whether the degrees of freedom for unbiased estimates should be based on numbers of cases ("count"; default) or sums of weights ("sum_wts").
Details

The weighted mean is computed as

$$\bar{x}_w = \frac{\sum_{i=1}^{k} x_i w_i}{\sum_{i=1}^{k} w_i}$$

where $x$ is a numeric vector and $w$ is a vector of weights.

The weighted variance is computed as

$$\text{var}_w(x) = \frac{\sum_{i=1}^{k} (x_i - \bar{x}_w)^2 w_i}{\sum_{i=1}^{k} w_i}$$

and the unbiased weighted variance is estimated by multiplying $\text{var}_w(x)$ by $\frac{k}{k-1}$.

Value

A weighted mean and variance if weights are supplied or an unweighted mean and variance if weights are not supplied.

Examples

```r
wt_dist(x = c(.1, .3, .5), wt = c(100, 200, 300))
wt_mean(x = c(.1, .3, .5), wt = c(100, 200, 300))
wt_var(x = c(.1, .3, .5), wt = c(100, 200, 300))
```
Index

* datagen
  simulate_d_database, 192
  simulate_psych, 197
  simulate_r_database, 199
  simulate_r_sample, 202
* datasets
  data_d_bb_multi, 69
  data_d_meas_multi, 69
  data_r_bvdr, 70
  data_r_bvrr, 70
  data_r_gonzalezmu, 2014, 71
  data_r_mcdaniel_1994, 71
  data_r_mcleod_2007, 72
  data_r_meas, 72
  data_r_meas_multi, 73
  data_r_oh_2009, 73
  data_r_roth_2015, 74
  data_r_uvdrr, 75
  data_r_uvdr, 75
* distribution
  simulate_alpha, 191
  simulate_matrix, 196
* output functions
  generate_bib, 115
  metabulate, 167
  metabulate_rmd_helper, 171
* regression
  metareg, 173
* univar
  mix_dist, 174
  truncate_dist, 206
  truncate_mean, 207
  truncate_var, 207
  wt_dist, 226
* utilities
  generate_directory, 117
  .ma_r_order2, 7
  .tau_squared_m_solver, 7
  .tau_squared_r_solver, 8
  adjust_n_d, 9
  adjust_n_r, 10
  anova.ma_psychmeta, 11
  composite_d_matrix, 12
  composite_d_scalar, 13
  composite_r_matrix, 12, 16
  composite_r_scalar, 14, 17
  composite_rel_matrix, 14
  composite_rel_scalar, 15
  composite_u_matrix, 19
  composite_u_scalar, 20
  compute_alpha, 21
  compute_dmod, 21
  compute_dmod_np, 24
  compute_dmod_par, 27
  conf.limits.nc.chisq, 29
  confidence, 31
  confidence_r, 32
  confint, 32
  control_intercor, 33
  control_psychmeta, 35, 133, 149
  convert_es, 5, 38
  convert_ma, 6, 39
  convert_meta(convert_ma), 39
  correct_d, 5, 41
  correct_d_bias, 43
  correct_d_bias(), 213
  correct_glass_bias, 45
  correct_matrix_mvrr, 46
  correct_means_mvrr, 47
  correct_r, 5, 48
  correct_r(), 41
  correct_r_bias, 52
  correct_r_bias(), 217, 223
  correct_r_coarseness, 53
  correct_r_dich, 54
  correct_r_split, 55
  create_ad, 5, 56, 87, 153
  create_ad_group, 62
228
create_ad_list(create_ad_tibble), 64
credibility, 68
data_d_bb_multi, 69
data_d_meas_multi, 6, 69
data_r_bvdrr, 6, 70
data_r_bvirr, 6, 70
data_r_gonzalezmule_2014, 5, 71
data_r_mcdaniel_1994, 5, 71
data_r_mcleod_2007, 72
data_r_meas, 6, 72
data_r_meas_multi, 6, 73
data_r_oh_2009, 73
data_r_roth_2015, 5, 74
data_r_uvdrr, 6, 75
data_r_uvirr, 6, 75
estimate_artifacts, 76
estimate_cor_prods(estimate_prod), 81
estimate_cov_prods(estimate_prod), 81
estimate_length sb, 80
estimate_mean_prod(estimate_prod), 81
estimate_q_dist, 83
estimate_rel_dist, 83
estimate_rel sb, 84
estimate_rxxa(estimate_artifacts), 76
estimate_rxxa_u(estimate_artifacts), 76
estimate_rxxi(estimate_artifacts), 76
estimate_rxxi_u(estimate_artifacts), 76
estimate_ryya(estimate_artifacts), 76
estimate_ryyi(estimate_artifacts), 76
estimate_u, 85
estimate_up(estimate_artifacts), 76
estimate_ut(estimate_artifacts), 76
estimate_ux(estimate_artifacts), 76
estimate_uy(estimate_artifacts), 76
estimate_var_artifacts, 87
estimate_var_prod(estimate_prod), 81
estimate_var_qxa
   (estimate_var_artifacts), 87
estimate_var_qxi
   (estimate_var_artifacts), 87
estimate_var_qya
   (estimate_var_artifacts), 87
estimate_var_qyi
   (estimate_var_artifacts), 87
estimate_var_rho_int, 97
estimate_var_rho_int_bvdrr
   (estimate_var_rho_int), 97
estimate_var_rho_int_bvirr
   (estimate_var_rho_int), 97
estimate_var_rho_int_meas
   (estimate_var_rho_int), 97
estimate_var_rho_int_rb
   (estimate_var_rho_int), 97
estimate_var_rho_int_ruvdrr
   (estimate_var_rho_int), 97
estimate_var_rho_int_ruvirr
   (estimate_var_rho_int), 97
estimate_var_rho_tsa
   (estimate_var_rho_tsa), 99
estimate_var_rho_tsa_bvdrr
   (estimate_var_rho_tsa), 99
estimate_var_rho_tsa_bvirr
   (estimate_var_rho_tsa), 99
estimate_var_rho_tsa_meas
   (estimate_var_rho_tsa), 99
estimate_var_rho_tsa_rb1
   (estimate_var_rho_tsa), 99
estimate_var_rho_tsa_rb2
   (estimate_var_rho_tsa), 99
estimate_var_rho_tsa_ruvdrr
   (estimate_var_rho_tsa), 99
estimate_var_rho_tsa_ruvirr
   (estimate_var_rho_tsa), 99
estimate_var_rxxa
   (estimate_var_rho_int), 97
estimate_var_rxxi
   (estimate_var_rho_int), 97
estimate_var_ryya
   (estimate_var_rho_int), 97
estimate_var_ryyi
   (estimate_var_rho_int), 97
estimate_var_rho_tsa
   (estimate_var_rho_tsa), 108
estimate_var_rho_tsa_bvdrr
   (estimate_var_rho_tsa), 108
estimate_var_rho_tsa_bvirr
   (estimate_var_rho_tsa), 108
estimate_var_rho_tsa_meas
   (estimate_var_rho_tsa), 108
estimate_var_rho_tsa_rb1
   (estimate_var_rho_tsa), 108
estimate_var_rho_tsa_rb2
   (estimate_var_rho_tsa), 108
estimate_var_rho_tsa_rb2()
   (estimate_var_rho_tsa), 111
estimate_var_rho_tsa_ruvdrr
(estimate_var_tsa), 108
estimate_var_tsa_uvirr
(estimate_var_tsa), 108
estimate_var_ut
(estimate_var_artifacts), 87
estimate_var_ux
(estimate_var_artifacts), 87
filter_ma, 112, 168, 181
filter_ma(), 11, 116, 178
filter_meta(filter_ma), 112
format_num, 113, 169
gamma function, 44, 45
generate_bib, 115, 169, 170, 172
generate_directory, 117
get_ad(get_stuff), 117
get_bootstrap(get_stuff), 117
get_cumulative(get_stuff), 117
get_escalc(get_stuff), 117
get_followup(get_stuff), 117
get_heterogeneity(get_stuff), 117
get_leave1out(get_stuff), 117
get_matrix(get_stuff), 117
get_metafor(get_stuff), 117
get_metareg(get_stuff), 117
get_metatab(get_stuff), 117
get_plots(get_stuff), 117
get_stuff, 117
ggplot2::facet_grid(), 178
heterogeneity, 6, 123
knitr::is_html_output, 172
knitr::is_latex_output, 172
limits_tau, 125
limits_tau2, 126
lm_mat, 127
lm_matrix(lm_mat), 127
ma_d, 5, 6, 130, 136, 153, 173
ma_d_ad, 134
ma_d_ad(ma_d), 130
ma_d_barebones(ma_d), 130
ma_d_bb(ma_d), 130
ma_d_ic(ma_d), 130
ma_d_order2, 143
ma_generic, 144
ma_r, 5, 6, 49, 58, 66, 77, 90, 135, 136, 146,
153, 173
ma_r(), 42, 216, 218, 221
ma_r_ad, 56, 149
ma_r_ad(ma_r), 146
ma_r_barebones(ma_r), 146
ma_r_bb(ma_r), 146
ma_r_ic(ma_r), 146
ma_r_order2, 6, 164
matreg(lm_mat), 127
matrixreg(lm_mat), 127
merge_simdat_d, 166, 192
merge_simdat_r, 166, 199
metabulate, 6, 116, 167, 171, 172
metabulate_rmd_helper, 116, 170, 171
metareg, 6, 173
mix_dist, 174
mix_matrix, 176
mix_r_2group, 177
output_format, 116, 168
plot_cefp(plot_funnel), 179
plot_forest, 178
plot_funnel, 179
predict, 182
print, 182
psychmeta(psychmeta-package), 4
psychmeta-package, 4
reattribute, 183
render, 169
reshape_mat2dat, 5, 184
reshape_vec2mat, 185
reshape_wide2long, 5, 187
rmarkdown::render(), 116
sensitivity, 6, 188
sensitivity_bootstrap(sensitivity), 188
sensitivity_cumulative(sensitivity),
188
sensitivity_leave1out(sensitivity), 188
simulate_alpha, 191
simulate_d_database, 6, 166, 192, 204
simulate_d_sample, 6, 195
simulate_matrix, 196
simulate_psych, 197
simulate_r_database, 6, 166, 199, 202, 205
simulate_r_sample, 6, 202
INDEX

sparsify_simdat_d, 192, 204
sparsify_simdat_r, 199, 205
stats::symnum(), 183
summary, 6, 205
tibble, 170
tibble::print.tbl(), 183
truncate_dist, 206
truncate_mean, 207
truncate_var, 207
unmix_matrix, 208
unmix_r_2group, 209
var_error_A, 210
var_error_alpha, 211
var_error_auc(var_error_A), 210
var_error_cles(var_error_A), 210
var_error_d, 212
var_error_delta, 213
var_error_g, 214
var_error_mult_R, 215
var_error_mult_Rsq(var_error_mult_R), 215
var_error_q, 216
var_error_R(var_error_mult_R), 215
var_error_r, 217
var_error_r_bvirr, 219
var_error_rel, 218
var_error_Rsq(var_error_mult_R), 215
var_error_spearman, 223
var_error_u, 224
wt_cor(wt_cov), 225
wt_cov, 225
wt_dist, 226
wt_mean(wt_dist), 226
wt_var(wt_dist), 226