Package: aggreCAT (via r-universe)

October 23, 2026

Type Package

Title Mathematically Aggregating Expert Judgments

Version 1.1.0

Description The use of structured elicitation to inform decision making has grown dramatically in recent decades, however, judgements from multiple experts must be aggregated into a single estimate. Empirical evidence suggests that mathematical aggregation provides more reliable estimates than enforcing behavioural consensus on group estimates. 'aggreCAT' provides state-of-the-art mathematical aggregation methods for elicitation data including those defined in Hanea, A. et al. (2021) <doi:10.1371/journal.pone.0256919>. The package also provides functions to visualise and evaluate the performance of your aggregated estimates on validation data.

URL https://replicats.research.unimelb.edu.au/, https://github.com/metamelb-repliCATS/aggreCAT, https://metamelb-replicats.github.io/aggreCAT/

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Suggests testthat (>= 2.1.0), knitr, rmarkdown, covr, pointblank, janitor, qualtRics, here, readxl, readr, lubridate, forcats, ggforce, ggpubr, ggridges, rjags, tidybayes, tidyverse, usethis, nlme, gt, gtExtras, R.rsp

RoxygenNote 7.3.3

Depends R (>= 3.5)

Imports magrittr, GoFKernel, purrr, R2jags, coda, precrec, mathjaxr, cli, VGAM, crayon, dplyr, stats, stringr, tidyr, tibble, ggplot2, insight, DescTools, MLmetrics

VignetteBuilder knitr, R.rsp

RdMacros mathjaxr

BuildManual TRUE

Config/testthat/parallel true

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

BugReports https://github.com/metamelb-repliCATS/aggreCAT/issues

Config/pak/sysreqs cmake make jags libicu-dev libuv1-dev libssl-dev libx11-dev zlib1g-dev

Repository https://metamelb-replicats.r-universe.dev

Date/Publication 2026-06-25 04:58:37 UTC

RemoteUrl https://github.com/metamelb-replicats/aggrecat

RemoteRef HEAD

RemoteSha e718ae9ff7cfc0156694bce3e2046527572000b7

Contents

AverageWAgg . . . 3
BayesianWAgg . . . 4
confidence_score_evaluation . . . 6
confidence_score_heatmap . . . 7
confidence_score_ridgeplot . . . 8
data_comments . . . 8
data_confidence_scores . . . 9
data_justifications . . . 10
data_outcomes . . . 10
data_ratings . . . 11
data_supp_priors . . . 11
data_supp_quiz . . . 12
data_supp_reasons . . . 12
DistributionWAgg . . . 13
ExtremisationWAgg . . . 15
IntervalWAgg . . . 17
LinearWAgg . . . 20
method_placeholder . . . 22
postprocess_judgements . . . 22
preprocess_judgements . . . 23
ReasoningWAgg . . . 24
ShiftingWAgg . . . 26
weight_asym . . . 28
weight_interval . . . 29
weight_nIndivInterval . . . 30
weight_outlier . . . 30
weight_reason . . . 31
weight_reason2 . . . 32
weight_varIndivInterval . . . 33

Index **34**

`AverageWAgg` *Aggregation Method: AverageWAgg*

Description

Calculate one of several types of averaged best estimates.

Usage

```
AverageWAgg(
  expert_judgements,
  type = "ArMean",
  name = NULL,
  placeholder = FALSE,
  percent_toggle = FALSE,
  round_2_filter = TRUE
)
```

Arguments

`expert_judgements`	A dataframe in the format of data_ratings.
`type`	One of `"ArMean"`, `"Median"`, `"GeoMean"`, `"LOArMean"`, or `"ProbitArMean"`.
`name`	Name for aggregation method. Defaults to `type` unless specified.
`placeholder`	Toggle the output of the aggregation method to impute placeholder data.
`percent_toggle`	Change the values to probabilities. Default is `FALSE`.
`round_2_filter`	Note that the IDEA protocol results in both a Round 1 and Round 2 set of probabilities for each claim. Unless otherwise specified, we will assume that the final Round 2 responses (after discussion) are being referred to.

Details

This function returns the average, median and transformed averages of best-estimate judgements for each claim.

`type` may be one of the following:

ArMean: Arithmetic mean of the best estimates

$$\hat{p}_c\left(ArMean\right) = \frac{1}{N}\sum_{i=1}^{N} B_{i,c}$$

Median: Median of the best estimates

$$\hat{p}_c\left(\text{median}\right) = \text{median} B^{i}_{ci=1,\ldots,N}$$

GeoMean: Geometric mean of the best estimates

$$GeoMean_c = \left(\prod_{i=1}^{N} B_{i,c} \right)^{\frac{1}{N}}$$

LOArMean: Arithmetic mean of the log odds transformed best estimates

$$LogOdds_{i,c} = \frac{1}{N} \sum_{i=1}^{N} log \left(\frac{B_{i,c}}{1 - B_{i,c}} \right)$$

The average log odds estimate is then back transformed to give a final group estimate:

$$\hat{p}_c \left(LOArMean \right) = \frac{e^{LogOdds_{i,c}}}{1 + e^{LogOdds_{i,c}}}$$

ProbitArMean: Arithmetic mean of the probit transformed best estimates

$$Probit_c = \frac{1}{N} \sum_{i=1}^{N} \Phi^{-1} \left(B_{i,c} \right)$$

The average probit estimate is then back transformed to give a final group estimate:

$$\hat{p}_c \left(ProbitArMean \right) = \Phi \left(Probit_c \right)$$

Value

A tibble of confidence scores `cs` for each `paper_id`.

Examples

```
AverageWAgg(data_ratings)
```

`BayesianWAgg` *Aggregation Method: BayesianWAgg*

Description

Bayesian aggregation methods with either uninformative or informative prior distributions.

JAGS Install

For instructions on installing JAGS onto your system visit https://gist.github.com/dennisprangle/e26923fae7477566510757ab3341f54c

Usage

```
BayesianWAgg(
  expert_judgements,
  type = "BayTriVar",
  priors = NULL,
  name = NULL,
  placeholder = FALSE,
  percent_toggle = FALSE,
  round_2_filter = TRUE
)
```

Arguments

`expert_judgements`	A dataframe in the format of data_ratings.
`type`	One of `"BayTriVar"`, or `"BayPRIORsAgg"`.
`priors`	(Optional) A dataframe of priors in the format of data_supp_priors, required for `type` `BayPRIORsAgg`.
`name`	Name for aggregation method. Defaults to `type` unless specified.
`placeholder`	Toggle the output of the aggregation method to impute placeholder data.
`percent_toggle`	Change the values to probabilities. Default is `FALSE`.
`round_2_filter`	Note that the IDEA protocol results in both a Round 1 and Round 2 set of probabilities for each claim. Unless otherwise specified, we will assume that the final Round 2 responses (after discussion) are being referred to.

Details

`type` may be one of the following:

BayTriVar: The Bayesian Triple-Variability Method, fit with JAGS.

Three kinds of variability around best estimates are considered:

1. generic claim variability: variation across individuals within a claim
2. generic participant variability: variation within an individual across claims
3. claim - participant specific uncertainty (operationalised by bounds): informed by interval widths given by individual i for claim c.

The model takes the log odds transformed individual best estimates as input (data), uses a normal likelihood function and derives a posterior distribution for the probability of replication.

$$log(\frac{B_{i,c}}{1-B_{i,c}}) \sim N(\mu_c, \sigma_{i,c}),$$

where μ_c denotes the mean estimated probability of replication for claim c, and $\sigma_{i,c}$ denotes the standard deviation of the estimated probability of replication for claim c and individual i (on the logit scale). Parameter $\sigma_{i,c}$ is calculated as:

$$\sigma_{i,c} = (U_{i,c} - L_{i,c} + 0.01) \times \sqrt{\sigma_i^2 + \sigma_c^2}$$

with σ_i denoting the standard deviation of estimated probabilities of replication for individual i and σ_c denoting the standard deviation of the estimated probability of replication for claim c.

The uninformative priors for specifying this Bayesian model are $\mu_c \sim N(0,\ 3)$, $\sigma_i \sim U(0,\ 10)$ and $\sigma_c \sim U(0,\ 10)$. After obtaining the median of the posterior distribution of μ_c, we can back transform to obtain $\hat{p}_c$:

$$\hat{p}_c\left(BayTriVar\right) = \frac{e^{\mu_c}}{1 + e^{\mu_c}}$$

BayPRIORsAgg: Priors derived from predictive models, updated with best estimates.

This method uses Bayesian updating to update a prior probability of replication estimated from a predictive model with an aggregate of the individuals' best estimates for any given claim. Methodology is the same as `type "BayTriVar"` except an informative prior is used for μ_c. Conceptually the parameters of the prior distribution of μ_c are informed by the PRIORS model (Gould et al. 2021) which is a multilevel logistic regression model that predicts the probability of replication using attributes of the original study. However, any model providing predictions of the probability of replication can be used to generate the required priors.

Value

A tibble of confidence scores `cs` for each `paper_id`.

Warning

Both `BayTriVar` and `BayPRIORsAgg` methods require a minimum of two claims for which judgements are supplied to `expert_judgements`. This is due to the mathematical definition of these aggregators: `BayesianWAgg` calculates the variance in best estimates across multiple claims as well as the variance in best estimates across claims per individual. Thus when only one claim is provided in `expert_judgements`, the variance is 0, hence more than one claim is required for the successful execution of both Bayesian methods.

Examples

```
## Not run: BayesianWAgg(data_ratings, percent_toggle = TRUE)
```

---

confidence_score_evaluation

Confidence Score Evaluation

Description

Evaluate the performance of the confidence scores generated by one or more aggregation methods. Assumes probabilistic confidence scores for the metrics selected.

Usage

```
confidence_score_evaluation(confidence_scores, outcomes)
```

Arguments

confidence_scores	A dataframe in the format output by the aggreCAT:: aggregation methods
outcomes	A dataframe with two columns: paper_id (corresponding to the id's from the confidence_scores), and outcome containing the known outcome of replication studies

Value

Evaluated dataframe with five columns: method (character variable describing the aggregation method), AUC (Area Under the Curve (AUC) scores of ROC curves - see ?precrec::auc), Brier_Score (see ?DescTools::BrierScore), Classification_Accuracy(classification accuracy measured for pcc = percent correctly classified; see ?MLmetrics::Accuracy), and Correlation (point-biserial correlation; see ?stats::cor).

Examples

```
confidence_score_evaluation(data_confidence_scores,
                            data_outcomes)
```

---

confidence_score_heatmap

Confidence Score Heat Map

Description

Confidence scores displayed on a colour spectrum across generated methods and assessed claims, split into predicted replication outcomes (TRUE/FALSE).

Usage

```
confidence_score_heatmap(confidence_scores = NULL, data_outcomes = NULL)
```

Arguments

confidence_scores	A data frame of confidence scores generated from the aggregation methods in the form of data_confidence_scores. Defaults to data_confidence_scores if no argument supplied.
data_outcomes	A data frame of unique claims and the associated binary outcome in the form of data_outcomes. If no argument supplied then defaults to data_outcomes supplied within package.

Value

Plot in viewer

Examples

```
confidence_score_heatmap(data_confidence_scores, data_outcomes)
```

`confidence_score_ridgeplot` *Confidence Score Ridge Plot*

Description

Display a ridge plot of confidence scores for each aggregation method

Usage

```
confidence_score_ridgeplot(confidence_scores = NULL)
```

Arguments

`confidence_scores`
: A data frame of confidence scores in long format in the form of data_confidence_scores

Value

A density ridge plot of aggregation methods

Examples

```
confidence_score_ridgeplot(data_confidence_scores)
```

`data_comments` *data_comments*

Description

data_comments

Usage

```
data_comments
```

Format

A tibble with 2 rows and 10 columns

round character string, both 'round_1' (before discussion)

paper_id character string identifying 2 unique papers

user_name factor for anonymized IDs for two participants

question character string for the type of question, both 'comprehension'

justification_id character string identifying 2 unique justifications

comment_id character string identifying 2 unique comments

commenter redundant column, same as user_name

comment character string with free-text response for the user

vote_count numeric, both 0

vote_sum numeric, both 0

group character string of group IDs that contained the participants

---

data_confidence_scores

Confidence Scores generated for 25 papers with 22 aggregation methods

---

Description

Confidence Scores generated for 25 papers with 22 aggregation methods

Usage

```
data_confidence_scores
```

Format

a tibble with 550 rows and 5 columns

method character string of method name

paper_id character string of paper IDs

cs numeric of generated confidence scores

n_experts numeric of the number of expert judgements aggregated in confidence score

data_justifications *Free-text justifications for expert judgements*

Description

Free-text justifications for expert judgements

Usage

```
data_justifications
```

Format

A table with 5630 rows and 9 columns:

round character string identifying whether the round was 1 (pre-discussion) or 2 (post-discussion)

paper_id character string of the paper ids (25 papers total)

user_name character string of anonymized IDs for each participant (25 participants included in this dataset)

question character string for the question type, with five options: flushing_freetext, involved_binary, belief_binary, direct_replication, and comprehension

justification character string with participant's free-text rationale for their responses

justification_id character string with a unique ID for each row

vote_count numeric of recorded votes (all 0 or 1)

vote_sum numeric of summed vote counts(all 0 or 1)

group character string of group IDs that contained the participants

data_outcomes *Replication outcomes for the papers*

Description

Replication outcomes for the papers

Usage

```
data_outcomes
```

Format

a tibble with 25 rows and 2 columns

paper_id character string for the paper ID

outcome numeric value of replication outcome. 1 = replication success, 0 = replication failure

data_ratings *P1_ratings*

Description

Anonymized expert judgements of known-outcome claims, assessed at the 2019 SIPS repliCATS workshop

Usage

```
data_ratings
```

Format

A table with 6880 rows and 7 columns:

round character string identifying whether the round was 1 (pre-discussion) or 2 (post-discussion)

paper_id character string of the claim ids (25 unique claims total)

user_name character string of anonymized IDs for each participant (25 participants included in this dataset)

question character string for the question type, with four options: direct_replication, involved_binary, belief_binary, or comprehension

element character string for the type of response coded in the row, with five options: three_point_lower, three_point_best, three_point_upper, binary_question, or likert_binary

value numeric value for the participant's response

group character string of group IDs that contained the participants

data_supp_priors *A table of prior means, to be fed into the BayPRIORsAgg aggregation method*

Description

A table of prior means, to be fed into the BayPRIORsAgg aggregation method

Usage

```
data_supp_priors
```

Format

A tibble of 25 rows and 2 columns

paper_id character string with a unique id for each row corresponding to the assessed claim (from 125 papers total)

prior_means numeric with the average prior probability for the claim corresponding to the paper_id

data_supp_quiz *A table of scores on the quiz to assess prior knowledge, to be fed into the QuizWAgg aggregation method*

Description

A table of scores on the quiz to assess prior knowledge, to be fed into the QuizWAgg aggregation method

Usage

```
data_supp_quiz
```

Format

A tibble 19 rows and 2 columns

user_name factor for anonymized IDs for each participant

quiz_score numeric for the participant's score on the quiz (min of 0, max of 16, NA if no questions answered)

quiz_score_even numeric quiz scores where all questions are weighted equally (the original quiz score assigned only half marks for true/false questions)

quiz_score_stats numeric participant score based only on questions pertaining to statistics

data_supp_reasons *Categories of reasons provided by participants for their expert judgements*

Description

Categories of reasons provided by participants for their expert judgements

Usage

```
data_supp_reasons
```

Format

a tibble with 625 rows and 15 columns

paper_id character string for the paper ID

user_name character string for participant ID

RW04 Date of publication numeric; references to the date of publication, for example in relation to something being published prior to the 'replication crisis' within the relevant discipline, or a study being difficult to re-run now because of changes in social expectations.

RW15 Effect size numeric; any references to the effect size that indicate that the participant considered the size of the effect when assessing the claim. Don't use if the term "effect size" is used in unrelated ways, but err on the side of considering statements as relevant to the participant's assessment.

RW16 Interaction effect numeric; references to when the effect was an interaction effect (rather than a direct effect).

RW17 Interval or range measure for statistical uncertainty (CI, SD, etc) numeric; references to the inclusion, absence, or size of the uncertainty measure for a given effect.

RW18 Outside participants areas of expertise numeric; references to the claim under assessment being outside the participant's areas of expertise.

RW20 Plausibility numeric; references to the plausibility of the claim.

RW21 Population or subject characteristics (sampling practices) numeric; references to the characteristics of the sample population or subjects used in a study that affect the participant's assessment of the claim, including references to low response rate and any other questions or appreciation of the sampling practices.

RW22 Power adequacy and or sample size numeric; combines 2 nodes for references to the adequacy (or not) of the statistical power of the study &/or sample size.

RW32 Reputation numeric; references to the reputation of the journal/institute/author.

RW37 Revision statements numeric; .

RW42 Significance, statistical (p-value etc) numeric; references to a test of statistical significance for the claim as it impacts on the participant's assessment. This explicitly includes p-values, t-values, critical alpha and p-rep.

`DistributionWAgg` *Aggregation Method: DistributionWAgg*

Description

Calculate the arithmetic mean of distributions created with expert judgements. The aggregate is the median of the average distribution fitted on the individual estimates.

Usage

```
DistributionWAgg(
  expert_judgements,
  type = "DistribArMean",
  name = NULL,
  placeholder = FALSE,
  percent_toggle = FALSE,
  round_2_filter = TRUE
)
```

Arguments

`expert_judgements`	A dataframe in the format of data_ratings.
`type`	One of `"DistribArMean"` or `"TriDistribArMean"`.
`name`	Name for aggregation method. Defaults to `type` unless specified.
`placeholder`	Toggle the output of the aggregation method to impute placeholder data.
`percent_toggle`	Change the values to probabilities. Default is `FALSE`.
`round_2_filter`	Note that the IDEA protocol results in both a Round 1 and Round 2 set of probabilities for each claim. Unless otherwise specified, we will assume that the final Round 2 responses (after discussion) are being referred to.

Details

This method assumes that the elicited probabilities and bounds can be considered to represent participants' subjective distributions associated with relative frequencies (rather than unique events). That is to say that we considered that the lower bound of the individual per claim corresponds to the 5th percentile of their subjective distribution on the probability of replication, denoted $q_{5,i}$, the best estimate corresponds to the median, $q_{50,i}$, and the upper bound corresponds to the 95th percentile, $q_{95,i}$. With these three percentiles, we can fit parametric or non-parametric distributions and aggregate them rather than the (point) best estimates.

`type` may be one of the following:

DistribArMean: Applies a non-parametric distribution evenly across upper, lower and best estimates.

Using the three percentiles we can build the minimally informative non-parametric distribution that spreads the mass uniformly between the three percentiles.

$$F_i(x) = \begin{cases} 0, \text{ for } x < 0 \\ \dfrac{0.05}{q_{5,i}} \cdot x, \text{ for } 0 < x < q_{5,i} \\ \dfrac{0.45}{q_{50,i} - q_{5,i}} \cdot (x - q_{5,i}) + 0.05, \text{ for } q_{5,i} \leq x < q_{50,i} \\ \dfrac{0.45}{q_{95,i} - q_{50,i}} \cdot (x - q_{50,i}) + 0.5, \text{ for } q_{50,i} \leq x < q_{95,i} \\ \dfrac{0.05}{1 - q_{95,i}} \cdot (x - q_{95,i}) + 0.95, \text{ for } q_{95,i} \leq x < 1 \\ 1, \text{ for } x \geq 1. \end{cases}$$

Then take the average of all constructed distributions of participants for each claim:

$$AvDistribution = \frac{1}{N} \sum_{i=1}^{N} F_i(x),$$

and the aggregation is the median of the average distribution:

$$\hat{p}_c\left(DistribArMean\right) = AvDistribution^{-1}(0.5).$$

TriDistribArMean: Applies a triangular distribution to the upper, lower and best estimates.

A more restrictive fit with different assumptions about the elicited best estimates, upper and lower bounds. We can assume that the lower and upper bounds form the support of the distribution, and the best estimate corresponds to the mode.

$$F_i(x) = \begin{cases} 0, \text{ for } x < L_i \\ \dfrac{(x - L_i)^2}{(U_i - L_i)(B_i - L_i)}, \text{ for } L_i \leq x < B_i \\ 1 - \dfrac{(U_i - x)^2}{(U_i - L_i)(U_i - B_i)}, \text{ for } B_i \leq x < U_i \\ 1, \text{ for } x \geq U_i. \end{cases}$$

Then take the average of all constructed distributions of participants for each claim:

$$AvDistribution = \frac{1}{N} \sum_{i=1}^{N} F_i(x),$$

and the aggregation is the median of the average distribution:

$$\hat{p}_c\left(TriDistribArMean\right) = AvDistribution^{-1}(0.5).$$

Value

A tibble of confidence scores `cs` for each `paper_id`.

Examples

```
DistributionWAgg(data_ratings, percent_toggle = TRUE)
```

`ExtremisationWAgg` *Aggregation Method: ExtremisationWAgg*

Description

Calculate beta-transformed arithmetic means of best estimates.

Usage

```
ExtremisationWAgg(
  expert_judgements,
  type = "BetaArMean",
  name = NULL,
  alpha = 6,
  beta = 6,
  cutoff_lower = NULL,
```

```
  cutoff_upper = NULL,
  placeholder = FALSE,
  percent_toggle = FALSE,
  round_2_filter = TRUE
)
```

Arguments

expert_judgements	A dataframe in the format of data_ratings.
type	One of "BetaArMean" or "BetaArMean2".
name	Name for aggregation method. Defaults to type unless specified.
alpha	parameter for the 'shape1' argument in the stats::pbeta function (defaults to 6)
beta	parameter for the 'shape2' argument in the stats::pbeta function (defaults to 6)
cutoff_lower	Lower bound of middle region without extremisation in "BetaArMean2" aggregation types.
cutoff_upper	Upper bound of middle region without extremisation in "BetaArMean2" aggregation types.
placeholder	Toggle the output of the aggregation method to impute placeholder data.
percent_toggle	Change the values to probabilities. Default is FALSE.
round_2_filter	Note that the IDEA protocol results in both a Round 1 and Round 2 set of probabilities for each claim. Unless otherwise specified, we will assume that the final Round 2 responses (after discussion) are being referred to.

Details

This method takes the average of best estimates and transforms it using the cumulative distribution function of a beta distribution.

type may be one of the following:

BetaArMean: Beta transformation applied across the entire range of calculated confidence scores.

$$\hat{p}_c\left(\text{BetaArMean}\right) = H_{\alpha\beta}\left(\frac{1}{N}\sum_{i=1}^{N} B_{i,c}\right),$$

where $H_{\alpha\beta}$ is the cumulative distribution function of the beta distribution with parameters α and β, which default to 6 in the function.

The justification for equal parameters (the 'shape1' and 'shape2' arguments in the stats::pbeta function) are outlined in Satopää et al (2014) and the references therein (note that the method outlined in that paper is called a beta-transformed linear opinion pool). To decide on the default shape value of 6, we explored the data_ratings dataset with random subsets of 5 assessments per claim, which we expect to have for most of the claims assessed by repliCATS.

BetaArMean2: Beta transformation applied only to calculated confidence scores that are outside a specified middle range. The premise being that we don't extremise "fence-sitter" confidence scores.

$$\hat{p}_c\,(\text{BetaArMean2}) = \begin{cases} H_{\alpha\beta}\left(\frac{1}{N}\sum_{i=1}^{N} B_{i,c}\right), \text{ for } \frac{1}{N}\sum_{i=1}^{N} B_{i,c} < \textit{cutoff_lower} \\ \frac{1}{N}\sum_{i=1}^{N} B_{i,c}, \text{ for } \textit{cutoff_lower} \leq \frac{1}{N}\sum_{i=1}^{N} B_{i,c} \leq \textit{cutoff_upper} \\ H_{\alpha\beta}\left(\frac{1}{N}\sum_{i=1}^{N} B_{i,c}\right), \text{ for } \frac{1}{N}\sum_{i=1}^{N} B_{i,c} > \textit{cutoff_upper} \end{cases}$$

Value

A tibble of confidence scores `cs` for each `paper_id`.

Examples

```
ExtremisationWAgg(data_ratings, percent_toggle = TRUE)
```

`IntervalWAgg` *Aggregation Method: IntervalWAgg*

Description

Calculate one of several types of linear-weighted best estimates where the weights are dependent on the lower and upper bounds of three-point elicitation (interval widths).

Usage

```
IntervalWAgg(
  expert_judgements,
  type = "IntWAgg",
  name = NULL,
  placeholder = FALSE,
  percent_toggle = FALSE,
  round_2_filter = TRUE
)
```

Arguments

`expert_judgements`	A dataframe in the format of data_ratings.
`type`	One of `"IntWAgg"`, `"IndIntWAgg"`, `"AsymWAgg"`, `"IndIntAsymWAgg"`, `"VarIndIntWAgg"`, `"KitchSinkWAgg"`.
`name`	Name for aggregation method. Defaults to `type` unless specified.
`placeholder`	Toggle the output of the aggregation method to impute placeholder data.
`percent_toggle`	Change the values to probabilities. Default is `FALSE`.

`round_2_filter` Note that the IDEA protocol results in both a Round 1 and Round 2 set of probabilities for each claim. Unless otherwise specified, we will assume that the final Round 2 responses (after discussion) are being referred to.

Details

The width of the interval provided by individuals may be an indicator of certainty, and arguably of accuracy of the best estimate contained between the bounds of the interval.

`type` may be one of the following:

IntWAgg: Weighted according to the interval width across individuals for that claim, rewarding narrow interval widths.

$$w_Interval_{i,c} = \frac{1}{U_{i,c} - L_{i,c}}$$

$$\hat{p}_c(IntWAgg) = \sum_{i=1}^{N} \tilde{w}_Interval_{i,c} B_{i,c}$$

where $U_{i,d} - L_{i,d}$ are individual i's judgements for claim d. Then

IndIntWAgg: Weighted by the rescaled interval width (interval width relative to largest interval width provided by that individual)

Because of the variability in interval widths between individuals across claims, it may be beneficial to account for this individual variability by rescaling interval widths across all claims per individual. This results in a re-scaled interval width weight, for individual i for claim c, relative to the widest interval provided by that individual across all claims C:

$$w_nIndivInterval_{i,c} = \frac{1}{\frac{U_{i,c} - L_{i,c}}{max((U_{i,d} - L_{i,d}) : d = 1, \ldots, C)}}$$

$$\hat{p}_c\left(IndIntWAgg\right) = \sum_{i=1}^{N} \tilde{w}_nIndivInterval_{i,c} B_{i,c}$$

AsymWAgg: Weighted by the asymmetry of individuals' intervals, rewarding increasing asymmetry.

We use the asymmetry of an interval relative to the corresponding best estimate to define the following weights:

$$w_asym_{i,c} = \begin{cases} 1 - 2 \cdot \frac{U_{i,c} - B_{i,c}}{U_{i,c} - L_{i,c}}, \text{for } B_{i,c} \geq \frac{U_{i,c} - L_{i,c}}{2} + L_{i,c} \\ 1 - 2 \cdot \frac{B_{i,c} - L_{i,c}}{U_{i,c} - L_{i,c}}, \text{otherwise} \end{cases}$$

then,

$$\hat{p}_c(AsymWAgg) = \sum_{i=1}^{N} \tilde{w}_asym_{i,c} B_{i,c}.$$

IndIntAsymWAgg: Weighted by individuals' interval widths and asymmetry

This rewards both asymmetric and narrow intervals. We simply multiply the weights calculated in the "AsymWAgg" and "IndIntWAgg" methods.

$$w_nIndivInterval_asym_{i,c} = \tilde{w}_nIndivInterval_{i,c} \cdot \tilde{w}_asym_{i,c}$$

$$\hat{p}_c(IndIntAsymWAgg) = \sum_{i=1}^{N} \tilde{w}_nIndivInterval_asym_{i,c} B_{i,c}$$

VarIndIntWAgg: Weighted by the variation in individuals' interval widths

A higher variance in individuals' interval width across claims may indicate a higher responsiveness to the supporting evidence of different claims. Such responsiveness might be predictive of more accurate assessors. We define:

$$w_varIndivInterval_i = var(U_{i,c} - L_{i,c}) : c = 1, \ldots, C,$$

where the variance (var) is calculated across all claims for individual i. Then,

$$\hat{p}_c(VarIndIntWAgg) = \sum_{i=1}^{N} \tilde{w}_varIndivInterval_i B_{i,c}$$

KitchSinkWAgg: Weighted by everything but the kitchen sink

This method is informed by the intuition that we want to reward narrow and asymmetric intervals, as well as the variability of individuals' interval widths (across their estimates). Again, we multiply the weights calculated in the "AsymWAgg", "IndIntWAgg" and "VarIndIntWAgg" methods above.

$$w_kitchSink_{i,c} = \tilde{w}_nIndivInterval_{i,c} \cdot \tilde{w}_asym_{i,c} \cdot \tilde{w}_varIndivInterval_i$$

$$\hat{p}_c(KitchSinkWAgg) = \sum_{i=1}^{N} \tilde{w}_kitchSink_{i,c} B_{i,c}$$

Value

A tibble of confidence scores cs for each paper_id.

Examples

```
IntervalWAgg(data_ratings)
```

`LinearWAgg` *Aggregation Method: LinearWAgg*

Description

Calculate one of several types of linear-weighted best estimates.

Usage

```
LinearWAgg(
  expert_judgements,
  type = "DistLimitWAgg",
  weights = NULL,
  name = NULL,
  placeholder = FALSE,
  percent_toggle = FALSE,
  flag_loarmean = FALSE,
  round_2_filter = TRUE
)
```

Arguments

`expert_judgements`	A dataframe in the format of data_ratings.
`type`	One of `"Judgement"`, `"Participant"`, `"DistLimitWAgg"`, `"GranWAgg"`, or `"OutWAgg"`.
`weights`	(Optional) A two column dataframe (`user_name` and `weight`) for `type = "Participant"` or a three two column dataframe (`paper_id', 'user_name` and `weight`) for `type = "Judgement"`
`name`	Name for aggregation method. Defaults to `type` unless specified.
`placeholder`	Toggle the output of the aggregation method to impute placeholder data.
`percent_toggle`	Change the values to probabilities. Default is `FALSE`.
`flag_loarmean`	A toggle to impute log mean (defaults `FALSE`).
`round_2_filter`	Note that the IDEA protocol results in both a Round 1 and Round 2 set of probabilities for each claim. Unless otherwise specified, we will assume that the final Round 2 responses (after discussion) are being referred to.

Details

This function returns weighted linear combinations of the best-estimate judgements for each claim.

`type` may be one of the following:

Judgement: Weighted by user-supplied weights at the judgement level

$$\hat{p}_c\left(JudgementWeights\right) = \sum_{i=1}^{N} \tilde{w}_judgement_{i,c} B_{i,c}$$

Participant: Weighted by user-supplied weights at the participant level

$$\hat{p}_c\left(ParticipantWeights\right) = \sum_{i=1}^{N} \tilde{w}_participant_i B_{i,c}$$

DistLimitWAgg: Weighted by the distance of the best estimate from the closest certainty limit. Giving greater weight to best estimates that are closer to certainty limits may be beneficial.

$$w_distLimit_{i,c} = \max\left(B_{i,c}, 1 - B_{i,c}\right)$$

$$\hat{p}_c\left(DistLimitWAgg\right) = \sum_{i=1}^{N} \tilde{w}_distLimit_{i,c} B_{i,c}$$

GranWAgg: Weighted by the granularity of best estimates

Individuals are weighted by whether or not their best estimates are more granular than a level of 0.05 (i.e., not a multiple of 0.05).

$$w_gran_i = \frac{1}{C} \sum_{d=1}^{C} \left\lceil \frac{B_{i,d}}{0.05} - \left\lfloor \frac{B_{i,d}}{0.05} \right\rfloor \right\rceil,$$

where $\lfloor\ \rfloor$ and $\lceil\ \rceil$ are the mathematical floor and ceiling functions respectively.

$$\hat{p}_c\left(GranWAgg\right) = \sum_{i=1}^{N} \tilde{w}_gran_i B_{i,c}$$

OutWAgg: Down weighting outliers

This method down-weights outliers by using the differences from the central tendency (median) of an individual's best estimates.

$$d_{i,c} = \left(medianB_{i,c}{}_{i=1,\ldots,N} - B_{i,c}\right)^2$$

$$w_out_i = 1 - \frac{d_{i,c}}{\max(d_c)})$$

$$\hat{p}_c\left(OutWAgg\right) = \sum_{i=1}^{N} \tilde{w}_out_i B_{i,c}$$

Value

A tibble of confidence scores `cs` for each `paper_id`.

Examples

```
LinearWAgg(data_ratings, percent_toggle = TRUE)
```

method_placeholder *Placeholder function with TA2 output*

Description

This functions stands in for when we haven't completed coding the method.

Usage

```
method_placeholder(expert_judgements, method_name)
```

Arguments

expert_judgements	A data frame in the form of ratings
method_name	Aggregation method to place into placeholder mode

Details

This function expects input from preprocess_judgements and outputs for postprocess_judgements.

Value

A tibble of confidence scores cs for each paper_id.

Examples

```
## Not run: method_placeholder(data_ratings, method_name = "TestMethod")
```

postprocess_judgements *Post-processing.*

Description

Standardise the output from aggregation method's. This function is called by every aggregation method as a final step.

Usage

```
postprocess_judgements(method_output)
```

Arguments

method_output	tibble created from one of the aggregation methods after pre-processing' with columns for the aggregation method, paper_id, aggregated_judgement and n_experts

Value

A tibble of confidence scores cs for each paper_id, corresponding with an aggregation method (character).

---

preprocess_judgements *Pre-process the data*

Description

Process input data with filters and meaningful variable names.

This function is called at the head of every aggregation method function.

Usage

```
preprocess_judgements(
  expert_judgements,
  round_2_filter = TRUE,
  three_point_filter = TRUE,
  percent_toggle = FALSE
)
```

Arguments

expert_judgements	A dataframe with the same variables (columns) as data_ratings.
round_2_filter	Note that the IDEA protocol results in both a Round 1 and Round 2 set of probabilities for each claim. Unless otherwise specified, we will assume that the final Round 2 responses (after discussion) are being referred to.
three_point_filter	Defaults TRUE to filter three point estimates. FALSE will filter the involved_binary question.
percent_toggle	Change the values to probabilities from percentages. Default is FALSE

Details

This pre-processing function takes input data in the format of data_ratings and outputs a dataframe that:

1. Applies any filters or manipulations required by the aggregation method.
2. Converts the input data into variables with more meaningful names for coding, to avoid errors in the wrangling process.

Value

a long tibble of expert judgements, with six columns: `round`, `paper_id`, `user_name`, `element` (i.e. question type), and `value` (i.e. participant response).

Examples

```
preprocess_judgements(data_ratings)
```

`ReasoningWAgg` *Aggregation Method: ReasoningWAgg*

Description

Calculate one of several types of linear-weighted best estimates using supplementary participant reasoning data to create weights.

Usage

```
ReasoningWAgg(
  expert_judgements,
  reasons = NULL,
  type = "ReasonWAgg",
  name = NULL,
  beta_transform = FALSE,
  beta_param = c(6, 6),
  placeholder = FALSE,
  percent_toggle = FALSE,
  flag_loarmean = FALSE,
  round_2_filter = TRUE
)
```

Arguments

`expert_judgements`	A dataframe in the format of data_ratings.
`reasons`	A dataframe in the form of data_supp_reasons
`type`	One of `"ReasonWAgg"`, `"ReasonWAgg2"`.
`name`	Name for aggregation method. Defaults to `type` unless specified.
`beta_transform`	Toggle switch to extremise confidence scores with the beta distribution. Defaults to `FALSE`.
`beta_param`	Length two vector of alpha and beta parameters of the beta distribution. Defaults to `c(6,6)`.
`placeholder`	Toggle the output of the aggregation method to impute placeholder data.
`percent_toggle`	Change the values to probabilities. Default is `FALSE`.

`flag_loarmean` A toggle to impute LOArMean instead of ArMean when no participants have a reasoning weight for a specific claim (defaults `FALSE`).

`round_2_filter` Note that the IDEA protocol results in both a Round 1 and Round 2 set of probabilities for each claim. Unless otherwise specified, we will assume that the final Round 2 responses (after discussion) are being referred to.

Details

Weighted by the breadth of reasoning provided to support the individuals' estimate.

`type` may be one of the following:

ReasonWAgg: Weighted by the number of supporting reasons

Giving greater weight to best estimates that are accompanied by a greater number of supporting reasons may be beneficial. We will consider $w_reason_{i,c}$ to be the number of unique reasons provided by that individual i in support of their estimate for claim c.

$$\hat{p}_c(ReasonWAgg) = \sum_{i=1}^{N} \tilde{w}_reason_{i,c} B_{i,c}$$

See Hanea et al. (2021) for an example of reason coding.

ReasonWAgg2: Incorporates both the number of reasons and their diversity across claims.

The claim diversity component of this score is calculated per individual from all claims they assessed. We assume each individual answers at least two claims. If an individual has assessed only one claim, there weighting for that claim is equivalent to "ReasonWAgg".

We will consider $w_varReason_{i,c}$ to be the weighted "number of unique reasons" provided by participant i in support of their estimate for claim c. Assume there are R total unique reasons any participant can use to justify their numerical answers. Then, for each participant i we can construct a matrix $\mathbf{CR_i}$ with R columns, each corresponding to a unique reason, r, and C rows, where C is the number of claims assessed by that participant. Each element of this matrix $\mathbf{CR_i}(r, c)$ can be either 1 or 0. $\mathbf{CR_i}(r, c) = 1$ if reason R_r was used to justify the estimates assessed for c, and $\mathbf{CR_i}(r, c) = 0$ if reason R_r was not mentioned when assessing claim c. The more frequently that a participant uses a given reason reduces the amount it contributes to the weight assigned to that participant.

$$w_varReason_{i,c} = \sum_{r=1}^{R} \mathbf{CR_i}(c, r) \cdot (1 - \frac{\sum_{c=1}^{C} \mathbf{CR_i}(c, r)}{C})$$

$$\hat{p}_c(ReasonWAgg2) = \sum_{i=1}^{N} \tilde{w}_varReason_{i,c} B_{i,c}$$

Value

A tibble of confidence scores `cs` for each `paper_id`.

Note

When `flag_loarmean` is set to `TRUE`, two additional columns will be returned; `method_applied` (a character variable describing the method actually applied with values of either `LoArMean` or `ReasonWAgg`) and `no_reason_score` (a logical variable describing whether no reasoning scores were supplied for any user for the given claim, where `TRUE` indicates no reasoning scores supplied and `FALSE` indicates that at least one participant for that claim had a reasoning score greater than 0).

named method_applied (with values LoArMean or ReasonWAgg), and no_reason_score, a logical variable describing whether or not there were no reasoning scores for that claim.

Examples

```
ReasoningWAgg(data_ratings, data_supp_reasons)
```

`ShiftingWAgg` *Aggregation Method: ShiftingWAgg*

Description

Weighted by judgements that shift the most after discussion

Usage

```
ShiftingWAgg(
  expert_judgements,
  type = "ShiftWAgg",
  name = NULL,
  placeholder = FALSE,
  percent_toggle = FALSE
)
```

Arguments

`expert_judgements`	A dataframe in the format of data_ratings.
`type`	One of `"ShiftWAgg"`, `"BestShiftWAgg"`, `"IntShiftWAgg"`, `"DistShiftWAgg"`, or `"DistIntShiftWAgg"`.
`name`	Name for aggregation method. Defaults to `type` unless specified.
`placeholder`	Toggle the output of the aggregation method to impute placeholder data.
`percent_toggle`	Change the values to probabilities. Default is `FALSE`.

Details

When judgements are elicited using the IDEA protocol (or any other protocol that allows experts to revisit their original estimates), the second round of estimates may differ from the original first set of estimates an expert provides. Greater changes between rounds will be given greater weight.

`type` may be one of the following:

ShiftWAgg: Takes into account the shift in all three estimates

Considers shifts across lower, $L_{i,c}$, and upper, $U_{i,c}$, confidence limits, and the best estimate, $B_{i,c}$. More emphasis is placed on changes in the best estimate such that:

$$w_Shift_{i,c} = |B1_{i,c} - B_{i,c}| + \frac{|L1_{i,c} - L_{i,c}| + |U1_{i,c} - U_{i,c}|}{2},$$

where $L1_{i,c}, B1_{i,c}, U1_{i,c}$ are the first round lower, best and upper estimates (prior to discussion) and $L_{i,c}, B_{i,c}, U1_{i,c}$ are the individual's revised second round estimates (after discussion).

$$\hat{p}_c(ShiftWAgg) = \sum_{i=1}^{N} \tilde{w}_Shift_{i,c} B_{i,c}$$

BestShiftWAgg: Weighted according to shifts in best estimates alone

Taking into account the fact that the scales best estimates are measured on are bounded, we can calculate shifts relative to the largest possible shift.

$$w_BestShift_{i,c} = \begin{cases} \frac{|B1_{i,c} - B_{i,c}|}{B1_{i_c}}, & for\ (B1_{i,c} > 0.5\ and\ B_{i,c} \leq 0.5) \\ & or\ B_{i,c} < B1_{i,c} \leq 0.5\ or\ B1_{i,c} > B_{i,c} > 0.5 \\ \frac{|B1_{i,c} - B_{i,c}|}{1 - B1_{i_c}}, & for\ (B1_{i,c} < 0.5\ and\ B_{i,c} \geq 0.5) \\ & or\ B1_{i,c} < B_{i,c} < 0.5\ or\ B_{i,c} > B1_{i,c} > 0.5. \end{cases}$$

$$\hat{p}_c(BestShiftWAgg) = \sum_{i=1}^{N} \tilde{w}_BestShift_{i,c} B_{i,c}$$

IntShiftWAgg: Weighted by shifts in interval widths alone.

Individuals whose interval widths narrow between rounds are given more weight.

$$w_IntShift_{i,c} = \frac{1}{(U_{i,c} - L_{i,c}) - (U1_{i,c} - L1_{i,c}) + 1}$$

$$\hat{p}_c(IntShiftWAgg) = \sum_{i=1}^{N} \tilde{w}_IntShift_{i,c} B_{i,c}$$

DistShiftWAgg: Weighted by whether best estimates become more extreme (closer to 0 or 1) between rounds.

$$w_DistShift_{i,c} = 1 - (\min(B_{i,c}, 1 - B_{i,c}) - \min(B1_{i,c}, 1 - B1_{i,c}))$$

$$\hat{p}_c(DistShiftWAgg) = \sum_{i=1}^{N} \tilde{w}_DistShift_{i,c} B_{i,c}$$

DistIntShiftWAgg: Rewards both narrowing of intervals and shifting towards the certainty limits between rounds.

We simply multiply the weights calculated in the "DistShiftWAgg" and "IntShiftWAgg" methods.

$$w_DistIntShift_{i,c} = \tilde{w}_IntShift_{i,c} \cdot \tilde{w}_DistShift_{i,c}$$

$$\hat{p}_c(DistIntShiftWAgg) = \sum_{i=1}^{N} \tilde{w}_DistIntShift_{i,c} B_{i,c}$$

Value

A tibble of confidence scores `cs` for each `paper_id`.

Examples

```
ShiftingWAgg(data_ratings)
```

`weight_asym` *Weighting method: Asymmetry of intervals*

Description

Calculates weights by asymmetry of intervals

Usage

```
weight_asym(expert_judgements)
```

Arguments

`expert_judgements`
: the long tibble exported from the `preprocess_judgements` function.

Details

This function is used inside IntervalWAgg to calculate the weights for the aggregation type `"AsymWAgg"`, `"IndIntAsymWAgg"` and `"KitchSinkWAgg"`. Pre-processed expert judgements (long format) are first converted to wide format then weighted by:

$$w_asym_{i,c} = \begin{cases} 1 - 2 \cdot \frac{U_{i,c} - B_{i,c}}{U_{i,c} - L_{i,c}}, \text{for } B_{i,c} \geq \frac{U_{i,c} - L_{i,c}}{2} + L_{i,c} \\ 1 - 2 \cdot \frac{B_{i,c} - L_{i,c}}{U_{i,c} - L_{i,c}}, \text{otherwise} \end{cases}$$

Data is converted back to long format, with only the weighted best estimates retained.

Value

A tibble in the form of the input `expert_judgements` argument with additional columns supplying the calculated weight for each row's observation.

See Also

Other weighting functions: `weight_interval()`, `weight_nIndivInterval()`, `weight_outlier()`, `weight_reason()`, `weight_reason2()`, `weight_varIndivInterval()`

Examples

```
weight_asym(preprocess_judgements(data_ratings))
```

---

weight_interval *Weighting method: Width of intervals*

Description

Calculates weights by interval width

Usage

```
weight_interval(expert_judgements)
```

Arguments

`expert_judgements`
: A dataframe in the form of data_ratings

Details

This function is used inside IntervalWAgg for aggregation type `"IntWAgg"`. It calculates the width of each three-point judgement (upper - lower), then returns the weight as the inverse of this interval.

Value

A tibble in the form of the input `expert_judgements` argument with additional columns supplying the calculated weight for each row's observation.

See Also

Other weighting functions: `weight_asym()`, `weight_nIndivInterval()`, `weight_outlier()`, `weight_reason()`, `weight_reason2()`, `weight_varIndivInterval()`

weight_nIndivInterval *Weighting method: Individually scaled interval widths*

Description

Weighted by the rescaled interval width within individuals across claims.

Usage

```
weight_nIndivInterval(expert_judgements)
```

Arguments

expert_judgements	A dataframe in the form of data_ratings

Details

This function is used inside IntervalWAgg for aggregation types "IndIntWAgg", "IndIntAsymWAgg" and "KitchSinkWAgg". Interval width weights are rescaled relative to an individuals interval widths across all claims.

$$w_nIndivInterval_{i,c} = \frac{1}{\frac{U_{i,c}-L_{i,c}}{\max((U_{i,d}-L_{i,d}):d=1,\ldots,C)}}$$

Value

A tibble in the form of the input expert_judgements argument with additional columns supplying the calculated weight for each row's observation.

See Also

Other weighting functions: weight_asym(), weight_interval(), weight_outlier(), weight_reason(), weight_reason2(), weight_varIndivInterval()

weight_outlier *Weighting method: Down weighting outliers*

Description

This method down-weights outliers.

Usage

```
weight_outlier(expert_judgements)
```

Arguments

`expert_judgements`
: A dataframe in the form of data_ratings

Details

This function is used by LinearWAgg to weighting functions for the aggregation type `"OutWAgg"`. Outliers are given less weight by using the squared difference between the median of an individual's best estimates across all claims and their best estimate for the claim being assessed:

$$d_{i,c} = \left(medianB_{i,c_{i=1,\ldots,N}} - B_{i,c}\right)^2$$

Weights are given by 1 minus the proportion of the individual's squared difference relative to the maximum squared difference for the claim across all individuals:

$$w_out_i = 1 - \frac{d_{i,c}}{\max(d_c)})$$

Value

A tibble in the form of the input `expert_judgements` argument with additional columns supplying the calculated weight for each row's observation.

See Also

Other weighting functions: `weight_asym()`, `weight_interval()`, `weight_nIndivInterval()`, `weight_reason()`, `weight_reason2()`, `weight_varIndivInterval()`

`weight_reason` *Weighting method: Total number of judgement reasons*

Description

This function is used by ReasoningWAgg to weighting functions for the aggregation type `"ReasonWAgg"`. Calculates weights based on the number of judgement reasoning methods used by an individual

Usage

```
weight_reason(expert_reasons)
```

Arguments

`expert_reasons` A dataframe in the form of data_supp_reasons

Details

Individuals' weight is equal to the maximum number of judgement reasons given

Value

A tibble of three columns `paper_id`, `user_name`, and `reason_count`

See Also

Other weighting functions: `weight_asym()`, `weight_interval()`, `weight_nIndivInterval()`, `weight_outlier()`, `weight_reason2()`, `weight_varIndivInterval()`

`weight_reason2` *Weighting method: Total number and diversity of judgement reasons*

Description

This function is used by ReasoningWAgg to weighting functions for the aggregation type `"ReasonWAgg2"`. Weights are based on the number and diversity of reasoning methods used by the participant to support their judgement.

Usage

```
weight_reason2(expert_reasons)
```

Arguments

`expert_reasons` A dataframe in the form of data_supp_reasons

Details

An individual's weight is a product of the number of reasons given in support of their judgement and the diversity of these reasons.

$$w_varReason_{i,c} = \sum_{r=1}^{R} \mathbf{CR_i}(c,r) \cdot (1 - \frac{\sum_{c=1}^{C} \mathbf{CR_i}(c,r)}{C})$$

Value

A tibble of three columns `paper_id`, `user_name`, and `reason_count`

See Also

Other weighting functions: `weight_asym()`, `weight_interval()`, `weight_nIndivInterval()`, `weight_outlier()`, `weight_reason()`, `weight_varIndivInterval()`

`weight_varIndivInterval` *Weighting method: Variation in individuals' interval widths*

Description

Calculates weights based on the variability of interval widths within individuals.

Usage

```
weight_varIndivInterval(expert_judgements)
```

Arguments

`expert_judgements`
: A dataframe in the form of data_ratings

Details

This function is used inside IntervalWAgg for aggregation types `"VarIndIntWAgg"` and `"KitchSinkWAgg"`. It calculates the difference between individual's upper and lower estimates, then calculates the variance in this interval across each individual's claim assessments.

$$w_varIndivInterval_i = var(U_{i,d} - L_{i,d}) : c = 1, ..., C$$

Value

A tibble in the form of the input `expert_judgements` argument with additional columns supplying the calculated weight for each row's observation.

See Also

Other weighting functions: `weight_asym()`, `weight_interval()`, `weight_nIndivInterval()`, `weight_outlier()`, `weight_reason()`, `weight_reason2()`

Index

∗ **datasets**
  data_comments, 8
  data_confidence_scores, 9
  data_justifications, 10
  data_outcomes, 10
  data_ratings, 11
  data_supp_priors, 11
  data_supp_quiz, 12
  data_supp_reasons, 12
∗ **weighting functions**
  weight_asym, 28
  weight_interval, 29
  weight_nIndivInterval, 30
  weight_outlier, 30
  weight_reason, 31
  weight_reason2, 32
  weight_varIndivInterval, 33

AverageWAgg, 3

BayesianWAgg, 4

confidence_score_evaluation, 6
confidence_score_heatmap, 7
confidence_score_ridgeplot, 8

data_comments, 8
data_confidence_scores, *7, 8*, 9
data_justifications, 10
data_outcomes, *7*, 10
data_ratings, *3*, *5*, 11, *14*, *16, 17*, *20*, *23, 24*, *26*, *29–31*, *33*
data_supp_priors, *5*, 11
data_supp_quiz, 12
data_supp_reasons, 12, *24*, *31, 32*
DistributionWAgg, 13

ExtremisationWAgg, 15

IntervalWAgg, 17, *28–30*, *33*

LinearWAgg, 20, *31*

method_placeholder, 22

postprocess_judgements, 22, *22*
preprocess_judgements, *22*, 23

ReasoningWAgg, 24, *31, 32*

ShiftingWAgg, 26

weight_asym, 28
weight_asym(), *29–33*
weight_interval, 29
weight_interval(), *29–33*
weight_nIndivInterval, 30
weight_nIndivInterval(), *29*, *31–33*
weight_outlier, 30
weight_outlier(), *29, 30*, *32, 33*
weight_reason, 31
weight_reason(), *29–33*
weight_reason2, 32
weight_reason2(), *29–33*
weight_varIndivInterval, 33
weight_varIndivInterval(), *29–32*