Package ‘metaggR’

October 13, 2022

Type Package

Title Calculate the Knowledge-Weighted Estimate

Version 0.3.0

Description According to a phenomenon known as ``the wisdom of the crowds,''
combining point estimates from multiple judges often provides a
more accurate aggregate estimate than using a point estimate from
a single judge. However, if the judges use shared information in
their estimates, the simple average will over-emphasize this common
component at the expense of the judges’ private information.
Asa Palley & Ville Satopää (2021) ``Boosting the Wisdom of Crowds
Within a Single Judgment Problem: Selective Averaging Based on Peer Predictions''
<https://papers.ssrn.com/sol3/Papers.cfm?abstract_id=3504286> proposes
a procedure for calculating a weighted average of the judges’ individual
estimates such that resulting aggregate estimate appropriately combines
the judges' collective information within a single estimation problem.
The authors use both simulation and data from six experimental studies
to illustrate that the weighting procedure outperforms existing averaging-like
methods, such as the equally weighted average, trimmed average, and median.
This aggregate estimate -- know as ``the knowledge-weighted estimate'' --
inputs a) judges' estimates of a continuous outcome (E) and
b) predictions of others' average estimate of this outcome (P).
In this R-package, the function knowledge_weighted_estimate(E,P)
implements the knowledge-weighted estimate. Its use is illustrated
with a simple stylized example and on real-world experimental data.

License GPL-2

Copyright (c) Ville Satopaa

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports MASS, stats

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 4.1)

NeedsCompilation no

Author Ville Satopää [aut, cre, cph],
Asa Palley [aut]

Maintainer Ville Satopää <`ville.satopaa@gmail.com`>

Repository CRAN

Date/Publication 2022-04-25 10:00:02 UTC

R topics documented:

`Calorie_Counts` *Data: Calorie Counts*

Description

Palley and Satopää (2021) conducted an experiment where participants were presented with 36 different pictures of food from different restaurants and were asked to estimate the total number of calories in these dishes. Each response involves three steps:

1. **Initial Estimates:** On the first screen the participant was presented with a picture of a meal and asked *How many calories do you think are in this meal?*
2. **Predictions of Others:** On the second screen the participant saw the same picture, was reminded of their previous estimate, and given the statement: *We will be showing this picture to other participants as well. Just as we did with you, we will ask them how many calories they believe are in this meal.* The participant was then asked to predict *How many calories do you think that others will guess on average?*
3. **Final Estimates:** On the third screen the participant saw the same picture again and was asked *After having reflected on others, what is your own final best estimate of the number of calories in this meal?*

Usage

```
E_CALORIES_INITIAL

E_CALORIES_FINAL

P_CALORIES

THETA_CALORIES

ID_CALORIES
```

Format

`E_CALORIES_INITIAL` is a list of the judges' initial estimates of the calorie counts in each of the 36 meals. Specifically, the jth element is a vector of the judges' initial estimates of the calories in the jth meal.

`E_CALORIES_FINAL` is a list of the judges' final estimates of the calorie counts in each of the 36 meals. Specifically, the jth element is a vector of the judges' final estimates of the calories in the jth meal.

`P_CALORIES` is a list of the judges' predictions of others. Specifically, the jth element is a vector of the judges' predictions of other judges' average estimate of the number of calories in the jth meal.

`THETA_CALORIES` is a vector of the true calorie counts in each of the 36 meals. Specifically, the jth element is the true calorie count in the jth meal.

`ID_CALORIES` is a list of the judges' identification numbers in each of the 36 meals. Specifically, the jth element is a vector of identification numbers of judges' who gave responses for the jth meal. These values make it possible to track a judge across questions.

Remark. The elements of each list correspond to the same meal. Specifically, the jth elements of `THETA_CALORIES`, `E_CALORIES_INITIAL`, `E_CALORIES_FINAL`, `P_CALORIES`, and `ID_CALORIES` represent the true calories, initial estimates, final estimates, the predictions of others, and identification numbers of the jth meal.

Source

Asa Palley and Ville Satopää. "Boosting the Wisdom of Crowds Within a Single Judgment Problem: Selective Averaging Based on Peer Predictions." https://papers.ssrn.com/sol3/Papers.cfm?abstract_id=3504286

`Coin_Flips` *Data: Coin Flips*

Description

Palley and Soll (2019) recruited individuals on Amazon Mechanical Turk and asked them to estimate the proportion of heads in 100 flips of a biased two-sided coin. The probability of heads was unknown to the participants, who were told that it could be anywhere between 1% and 99%. Before responding, each judge was shown a sample of flips that all judges saw (shared information) and another sample of flips that was only seen by that individual or by a subset of judges (private information). Three information structures were considered:

1. **Symmetric:** All judges saw their own unique sample of flips. There are a total of 72 judgment tasks under this condition.
2. **Nested:** Some judges saw only the shared sample while others saw an additional common sample. There are a total of 24 judgment tasks under this condition.
3. **Nested-Symmetric:** Some judges saw only the shared sample while others saw their own additional sample of flips. There are a total of 24 judgment tasks under this condition.

Usage

```
E_COINS_SYMMETRIC

E_COINS_NESTED

E_COINS_NESTED_SYMMETRIC

P_COINS_SYMMETRIC

P_COINS_NESTED

P_COINS_NESTED_SYMMETRIC

THETA_COINS_SYMMETRIC

THETA_COINS_NESTED

THETA_COINS_NESTED_SYMMETRIC

ID_COINS_SYMMETRIC

ID_COINS_NESTED

ID_COINS_NESTED_SYMMETRIC
```

Format

`E_COINS_SYMMETRIC` is a list of the judges' estimates of the proportion of heads in 100 flips of a biased two-sided coin under the Symmetric condition. Specifically, the jth element is a vector of the judges' estimated proportions in the jth task.

`E_COINS_NESTED` is a list of the judges' estimates of the proportion of heads in 100 flips of a biased two-sided coin under the Nested condition. Specifically, the jth element is a vector of the judges' estimated proportions in the jth task.

`E_COINS_NESTED_SYMMETRIC` is a list of the judges' estimates of the proportion of heads in 100 flips of a biased two-sided coin under the Nested-Symmetric condition. Specifically, the jth element is a vector of the judges' estimated proportions in the jth task.

`P_COINS_SYMMETRIC` is a list of the judges' predictions of other judges' average estimate of the proportion of heads in 100 flips of a biased two-sided coin under the Symmetric condition. Specifically, the jth element is a vector of the judges' predictions of others in the jth task.

`P_COINS_NESTED` is a list of the judges' predictions of other judges' average estimate of the proportion of heads in 100 flips of a biased two-sided coin under the Nested condition. Specifically, the jth element is a vector of the judges' predictions of others in the jth task.

`P_COINS_NESTED_SYMMETRIC` is a list of the judges' predictions of other judges' average estimate of the proportion of heads in 100 flips of a biased two-sided coin under the Nested-Symmetric condition. Specifically, the jth element is a vector of the judges' predictions of others in the jth task.

`THETA_COINS_SYMMETRIC` is a vector of the actual proportions of heads under the Symmetric condition. Specifically, the jth element is the actual proportion of heads in the jth task.

`THETA_COINS_NESTED` is a vector of the actual proportions of heads under the Nested condition. Specifically, the jth element is the actual proportion of heads in the jth task.

`THETA_COINS_NESTED_SYMMETRIC` is a vector of the actual proportions of heads under the Nested-Symmetric condition. Specifically, the jth element is the actual proportion of heads in the jth task.

`ID_COINS_SYMMETRIC` is a list of the judges' identification numbers in the judgment tasks under the Symmetric condition. Specifically, the jth element is a vector of identification numbers of judges' who participated in estimating the proportion of heads in the jth task. These values make it possible to track a judge across judgment tasks.

`ID_COINS_NESTED` is a list of the judges' identification numbers in the judgment tasks under the Nested condition. Specifically, the jth element is a vector of identification numbers of judges' who participated in estimating the proportion of heads in the jth task. These values make it possible to track a judge across judgment tasks.

`ID_COINS_NESTED_SYMMETRIC` is a list of the judges' identification numbers in the judgment tasks under the Nested-Symmetric condition. Specifically, the jth element is a vector of identification numbers of judges' who participated in estimating the proportion of heads in the jth task. These values make it possible to track a judge across judgment tasks.

Remark. The elements of each list correspond to the same meal. For instance, the jth elements of `THETA_COINS_SYMMETRIC`, `E_COINS_SYMMETRIC`, `P_COINS_SYMMETRIC`, and `ID_COINS_SYMMETRIC` represent the true proportion, estimates, the predictions of others, and identification numbers associated with the jth task under the Symmetric condition.

Source

Asa Palley and Jack Soll. "Extracting the Wisdom of Crowds When Information Is Shared." doi: 10.1287/mnsc.2018.3047

`General_Knowledge_Statements` *Data: General Knowledge Statements*

Description

Martinie et al. (2020) recruited individuals on Amazon Mechanical Turk and asked them to provide subjective probabilities of whether various general science statements from U.S. grade school were true or false. Problems were classified into five levels of difficulty, with level 1 being the easiest and level 5 being the most difficult. For example, one easy problem (level 1) presented the statement *Omnivores only eat meat*, whereas one difficult problem (level 5) presented the statement *Sound waves and electromagnetic waves are examples of longitudinal waves*.

The full data have been split into 5 groups based on the difficulty the questions.

1. `E_GK_1` to `E_GK_5`: A list of the judges' estimates of the probabilities that the statements are true.
2. `P_GK_1` to `P_GK_5`: A list of the judges' predictions of others' average probability estimates.
3. `ID_GK_1` to `ID_GK_5`: A list of the judges' identification numbers. These values make it possible to track a judge across different judgment tasks.
4. `THETA_GK_1` to `THETA_GK_5`: Actual outcomes showing whether the statements are true (1) or not (0).

The final number in the name of the data set indicates the associated difficulty level. For instance, `E_GK_5` holds the probability estimates of the most difficult questions, `THETA_GK_1` holds actual outcomes for the easiest questions, and so on. The elements of each list correspond to the same question. For instance, the jth elements of `THETA_GK_1`, `E_GK_1`, `P_GK_1`, and `ID_GK_1` give the true outcome, vector of probability estimates, vector of predictions of other judges' average probability estimates, and vector of identification numbers of the jth question with difficulty level 1.

Usage

```
E_GK_1

E_GK_2

E_GK_3
```

E_GK_4

E_GK_5

P_GK_1

P_GK_2

P_GK_3

P_GK_4

P_GK_5

THETA_GK_1

THETA_GK_2

THETA_GK_3

THETA_GK_4

THETA_GK_5

ID_GK_1

ID_GK_2

ID_GK_3

ID_GK_4

ID_GK_5

Format

E_GK_1 holds judges' estimates of the outcome. Specifically, it holds a list of 100 elements, one per general knowledge statement with difficulty level 1. The jth element is a vector of the judges' estimates of the probability that the jth statement is true.

E_GK_2 holds judges' estimates of the outcome. Specifically, it holds a list of 100 elements, one per general knowledge statement with difficulty level 2. The jth element is a vector of the judges' estimates of the probability that the jth statement is true.

E_GK_3 holds judges' estimates of the outcome. Specifically, it holds a list of 100 elements, one per general knowledge statement with difficulty level 3. The jth element is a vector of the judges' estimates of the probability that the jth statement is true.

E_GK_4 holds judges' estimates of the outcome. Specifically, it holds a list of 100 elements, one per general knowledge statement with difficulty level 4. The jth element is a vector of the judges' estimates of the probability that the jth statement is true.

`E_GK_5` holds judges' estimates of the outcome. Specifically, it holds a list of 100 elements, one per general knowledge statement with difficulty level 5. The jth element is a vector of the judges' estimates of the probability that the jth statement is true.

`P_GK_1` holds judges' predictions of other judges' average estimate of the outcome. Specifically, it holds a list of 100 elements, one per general knowledge statement with difficulty level 1. The jth element is a vector of the judges' predictions of others' average estimate of the probability that the jth statement is true.

`P_GK_2` holds judges' predictions of other judges' average estimate of the outcome. Specifically, it holds a list of 100 elements, one per general knowledge statement with difficulty level 2. The jth element is a vector of the judges' predictions of others' average estimate of the probability that the jth statement is true.

`P_GK_3` holds judges' predictions of other judges' average estimate of the outcome. Specifically, it holds a list of 100 elements, one per general knowledge statement with difficulty level 3. The jth element is a vector of the judges' predictions of others' average estimate of the probability that the jth statement is true.

`P_GK_4` holds judges' predictions of other judges' average estimate of the outcome. Specifically, it holds a list of 100 elements, one per general knowledge statement with difficulty level 4. The jth element is a vector of the judges' predictions of others' average estimate of the probability that the jth statement is true.

`P_GK_5` holds judges' predictions of other judges' average estimate of the outcome. Specifically, it holds a list of 100 elements, one per general knowledge statement with difficulty level 5. The jth element is a vector of the judges' predictions of others' average estimate of the probability that the jth statement is true.

`THETA_GK_1` is a vector of 100 elements, one per general knowledge statement with difficulty level 1. The jth element shows whether the jth general statement is true (1) or false (0).

`THETA_GK_2` is a vector of 100 elements, one per general knowledge statement with difficulty level 2. The jth element shows whether the jth general statement is true (1) or false (0).

`THETA_GK_3` is a vector of 100 elements, one per general knowledge statement with difficulty level 3. The jth element shows whether the jth general statement is true (1) or false (0).

`THETA_GK_4` is a vector of 100 elements, one per general knowledge statement with difficulty level 4. The jth element shows whether the jth general statement is true (1) or false (0).

`THETA_GK_5` is a vector of 100 elements, one per general knowledge statement with difficulty level 5. The jth element shows whether the jth general statement is true (1) or false (0).

`ID_GK_1` holds judges' identification numbers. Specifically, it holds a list of 100 elements, one per general knowledge statement with difficulty level 1. The jth element is a vector of numbers identifying the judges who provides responses for the jth statement. These values make it possible to track a judge across questions.

`ID_GK_2` holds judges' identification numbers. Specifically, it holds a list of 100 elements, one per general knowledge statement with difficulty level 2. The jth element is a vector of numbers identifying the judges who provides responses for the jth statement. These values make it possible to track a judge across questions.

`ID_GK_3` holds judges' identification numbers. Specifically, it holds a list of 100 elements, one per general knowledge statement with difficulty level 3. The jth element is a vector of numbers identifying the judges who provides responses for the jth statement. These values make it possible to track a judge across questions.

`ID_GK_4` holds judges' identification numbers. Specifically, it holds a list of 100 elements, one per general knowledge statement with difficulty level 4. The jth element is a vector of numbers identifying the judges who provides responses for the jth statement. These values make it possible to track a judge across questions.

`ID_GK_5` holds judges' identification numbers. Specifically, it holds a list of 100 elements, one per general knowledge statement with difficulty level 5. The jth element is a vector of numbers identifying the judges who provides responses for the jth statement. These values make it possible to track a judge across questions.

Source

Marcellin Martinie, Tom Wilkening, and Piers D. L. Howe. "Using meta-predictions to identify experts in the crowd when past performance is unknown" https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0232058

`get_influence_scores` *Calculate the Influence Scores*

Description

This function computes and plots the influence scores described in *Palley & Satopää (2021): Boosting the Wisdom of Crowds Within a Single Judgment Problem: Weighted Averaging Based on Peer Predictions*. The current version of the paper is available at https://papers.ssrn.com/sol3/Papers.cfm?abstract_id=3504286

Usage

```
get_influence_scores(E, P, plotIt = FALSE, cutoff = 7/2)
```

Arguments

`E`	Vector of $J \geq 6$ estimates of the outcome.
`P`	Vector of $J \geq 6$ predictions of others. The values must be in the same order as the estimates in `E`. Specifically, for all $j = 1, ..., J$, `E[j]` and `P[j]` give the jth judge's estimate and prediction of others, respectively.
`plotIt`	A boolean value. If TRUE, then the function call produces two side-by-side plots:

1. Left plot: This is a scatter plot of the judges' estimates against the judges' implied predictions of others. This plot includes regression lines both with (solid black) and without (dashed red) the exceptionally influential judges. All exceptionally influential judges are shown in red. The knowledge-weighted estimate is shown both with (black square) and without (red circle) exceptionally influential judges.
2. Right plot: This shows the judges' influence scores. All exceptionally influential judges are shown in red. The dashed horizontal line represents the threshold, defined as the user-defined `cutoff` value x the interquartile range of all influence scores.

For more information on the plots, see the Electronic Companion of *Palley & Satopää (2021): Boosting the Wisdom of Crowds Within a Single Judgment Problem: Weighted Averaging Based on Peer Predictions* at https://papers.ssrn.com/sol3/Papers.cfm?abstract_id=3504286.

`cutoff` A positive scalar describing the cutoff value for the outlier-robust knowledge-weighted estimate. The outlier-robust version calculates the influence scores for all judges. Each influence score is then compared against `cutoff` x the interquartile range of all influence scores. If a judge's influence score is above this quantity, then that judge is deemed exceptionally influential. This parameter only has an effect if `plotIt` has been set to TRUE.

Value

J vector of influence scores. Intuitively, the influence score of a judge represents the amount by which the knowledge-weighted estimate would change if that judge was removed from the crowd. Judges with an exceptionally high influence should be inspected. As a default cutoff value, the authors recommend $7/2$ times the interquartile range of the individual judges' influence scores.

Examples

```
# Illustration on the Three Gorges Dam Example in Palley & Satopää (2021):

# The original example with 6 judges is augmented with a 7th judge with an extreme response.
# Judges' estimates:
E2 = c(50, 134, 206, 290, 326, 374, 1000)
# Judges' predictions of others
P2 = c(26, 92, 116, 218, 218, 206, 400)

# The influence score of the 7th judge is much higher than the other judges' scores.
# This judge's response should be inspected and potentially excluded from
# the final knowledge-weighted estimate.
get_influence_scores(E2,P2)
```

`Grocery_Prices` *Data: Grocery Prices*

Description

Palley and Soll (2019) recruited volunteers passing through the student union to estimate the total price of 10 different bundles of nonperishable grocery items. Examples of items include a bottle of 190 Lil Critters Gummy Vites Sour Complete multivitamins ($10.93), a 5-oz. can of Wild Planet wild albacore tuna in extra virgin olive oil ($4.19), and an 11 oz. bag of Stauffer's Animal Crackers ($1.00).

Usage

```
E_GROCERIES

P_GROCERIES

THETA_GROCERIES

ID_GROCERIES
```

Format

`E_GROCERIES` is a list of the judges' estimates of the prices in each of the 10 bundles of groceries. Specifically, the jth element is a vector of the judges' estimates of the price of the jth bundle.

`P_GROCERIES` is a list of the judges' predictions of others. Specifically, the jth element is a vector of the judges' predictions of other judges' average estimate of the price of the jth bundle.

`THETA_GROCERIES` is a vector of the prices of the 10 bundles of groceries. Specifically, the jth element is the actual price of the jth bundle.

`ID_GROCERIES` is a list of the judges' identification numbers in the judgment tasks. Specifically, the jth element is a vector of identification numbers of judges' who participated in estimating the price of the jth bundle. These values make it possible to track a judge across judgment tasks.

Remark. The elements of each list correspond to the same judgment task. Specifically, the jth elements of `THETA_GROCERIES`, `E_GROCERIES`, `P_GROCERIES`, and `ID_GROCERIES` represent the true price, estimates, the predictions of others, and identification numbers associated with the jth bundle.

Source

Asa Palley and Jack Soll. "Extracting the Wisdom of Crowds When Information Is Shared." doi: 10.1287/mnsc.2018.3047

`knowledge_gap` *Calculate the Knowledge Gap*

Description

This function computes the knowledge gap described in *Palley & Satopää (2021): Boosting the Wisdom of Crowds Within a Single Judgment Problem: Weighted Averaging Based on Peer Predictions*. The current version of the paper is available at https://papers.ssrn.com/sol3/Papers.cfm?abstract_id=3504286

Usage

```
knowledge_gap(E, P, alpha)
```

Arguments

`E`	Vector of $J \geq 5$ estimates of the outcome.
`P`	Vector of $J \geq 5$ predictions of others. The values must be in the same order as the estimates in `E`. Specifically, for all $j = 1, ..., J$, `E[j]` and `P[j]` give the jth judge's estimate and prediction of others, respectively.
`alpha`	Vector of $J \geq 5$ weights. The `alpha[j]` element is the weight assigned to `E[j]`. The weights can be any values in the real line as long as they sum to 1.

Value

A singular value representing the knowledge gap. This represents the expected distance between the weighted combination of the judges' estimates, where the weights have been given by `alpha`, and the optimal aggregate estimate called the Global Posterior Expectation (GPE).

Examples

```
# Illustration on the Three Gorges Dam Example in Palley & Satopää (2021):

# Judges' estimates:
E = c(50, 134, 206, 290, 326, 374)
# Judges' predictions of others
P = c(26, 92, 116, 218, 218, 206)

# First find the knowledge-weights that minimize the knowledge gap:
alpha = knowledge_weights(E,P)
knowledge_gap(E,P, alpha)

# Small perturbations increase the knowledge gap:
alpha_per = alpha
alpha_per[1] = alpha_per[1] + 0.001
alpha_per[2] = alpha_per[2] - 0.001
knowledge_gap(E,P, alpha_per)
```

`knowledge_weighted_estimate` *Knowledge-Weighted Estimate*

Description

This function computes the knowledge-weighted estimate from *Palley & Satopää (2021): Boosting the Wisdom of Crowds Within a Single Judgment Problem: Weighted Averaging Based on Peer Predictions*. The current version of the paper is available at `https://papers.ssrn.com/sol3/Papers.cfm?abstract_id=3504286`.

Usage

```
knowledge_weighted_estimate(
  E,
  P,
  cutoff = 7/2,
  remove_inf = FALSE,
  no_inf_check = FALSE
)
```

Arguments

`E`	Vector of J estimates of the outcome. If influence scores are calculated (i.e., `no_inf_check` is FALSE), then the function call requires $J \geq 6$; else the knowledge-weighted estimated requires at least $J \geq 5$ judges.
`P`	Vector of J predictions of others. The values must be in the same order as the estimates in `E`. Specifically, for all $j = 1, ..., J$, `E[j]` and `P[j]` give the jth judge's estimate and prediction of others, respectively.
`cutoff`	A positive scalar describing the cutoff value for the outlier-robust knowledge-weighted estimate. The outlier-robust version calculates the influence scores for all judges (see `get_influence_scores` function). Each influence score is then compared against `cutoff` x the interquartile range of all influence scores. If a judge's influence score is above this quantity, then that judge is deemed exceptionally influential. By default, the influence scores are checked and a warning is given if an exceptionally influential judge is found. To turn off this feature, set `no_inf_check` to TRUE.
`remove_inf`	A boolean value. If TRUE, then all exceptionally influential judges are removed before the knowledge-weighted estimate is calculated. If FALSE, then the knowledge-weighted estimate is calculated based on the responses of all J judges.
`no_inf_check`	A boolean value. If TRUE, then the influence scores are not calculated at any point. This can be helpful to speed up calculations. However, the authors recommend checking for influential judges each time the knowledge weighted estimate is applied.

Value

A singular value representing the knowledge-weighted estimate

Examples

```
# Illustration on the Three Gorges Dam Example in Palley & Satopää (2021):

# Judges' estimates:
E1 = c(50, 134, 206, 290, 326, 374)
# Judges' predictions of others
P1 = c(26, 92, 116, 218, 218, 206)

# Knowledge-weighted estimate is 329.305
knowledge_weighted_estimate(E1,P1)

# The original example with 6 judges is augmented with a 7th judge with an extreme response.
# Judges' estimates:
E2 = c(50, 134, 206, 290, 326, 374, 1000)
# Judges' predictions of others
P2 = c(26, 92, 116, 218, 218, 206, 400)

# Knowledge-weighted estimate is 630.0491
# The function call warns of exceptionally influential judges.
knowledge_weighted_estimate(E2,P2)

# Calculate the knowledge-weighted estimate without influence score checking.
knowledge_weighted_estimate(E2,P2, no_inf_check = TRUE)

# Calculate the influence scores of the judges.
# This reveals that the 7th judge is highly influential.
get_influence_scores(E2,P2)

# Calculate the outlier-robust knowledge-weighted estimate.
# This removes all highly influential judges, namely judge 7 in this simple example,
# and returns the knowledge-weighted estimate of the remaining judges' estimates.
# This estimate aligns with the original 329.305
knowledge_weighted_estimate(E2,P2, remove_inf = TRUE)
```

---

knowledge_weights *Calculate the Weights that Minimize the Knowledge Gap*

Description

This function computes the weighted used in the knowledge-weighted estimate of *Palley & Satopää (2021): Boosting the Wisdom of Crowds Within a Single Judgment Problem: Weighted Averaging Based on Peer Predictions*. The current version of the paper is available at https://papers.ssrn.com/sol3/Papers.cfm?abstract_id=3504286

Usage

```
knowledge_weights(E, P)
```

Arguments

`E`	Vector of $J \geq 5$ estimates of the outcome.
`P`	Vector of $J \geq 5$ predictions of others. The values must be in the same order as the estimates in `E`. Specifically, for all $j = 1, ..., J$, `E[j]` and `P[j]` give the jth judge's estimate and prediction of others, respectively.

Value

$Jx1$ vector of weights that minimizes the knowledge gap and lead to the knowledge-weighted estimate.

Examples

```
# Illustration on the Three Gorges Dam Example in Palley & Satopää (2021):

# Judges' estimates:
E = c(50, 134, 206, 290, 326, 374)
# Judges' predictions of others
P = c(26, 92, 116, 218, 218, 206)

# Weights used in the knowledge-weighted estimate:
alpha = knowledge_weights(E,P)

# Knowledge-weighted estimate is 329.3266
t(alpha) %*% E

# Alternatively, the knowledge-weighted estimate can be calculated using
# the knowledge_weighted_estimate() function. This returns 329.305, which
# is slightly different from the above result. The difference arises because
# knowledge_weighted_estimate() improves stability by standardizing the
# judges' responses before aggregating them.
knowledge_weighted_estimate(E,P)
```

`NCAA_Basketball` *Data: NCAA Basketball*

Description

Palley and Soll (2019) recruited participants through ClearVoice Research and Amazon Mechanical Turk to estimate the probability that one team or the other would win various games in the 2014, 2015, and 2016 NCAA Division I Men's Basketball Tournaments. The responses for the Round of 64 games and Round of 16 games are treated separately because the Round of 64 games happen at the start of the tournament and often involve heavily mismatched teams (e.g., a 1 seed versus a 16 seed) while Round of 16 games typically involve more evenly matched teams, with implied betting market probabilities closer to 50%.

Usage

```
E_NCAA_R64

E_NCAA_R16

P_NCAA_R64

P_NCAA_R16

THETA_NCAA_R64

THETA_NCAA_R16

ID_NCAA_R64

ID_NCAA_R16
```

Format

`E_NCAA_R64` is a list of the judges' estimates of the probability that the given team wins in Round of 64. Specifically, the jth element is a vector of the judges' estimated probability in the jth game.

`E_NCAA_R16` is a list of the judges' estimates of the probability that the given team wins in Round of 16. Specifically, the jth element is a vector of the judges' estimated probability in the jth game.

`P_NCAA_R64` is a list of the judges' predictions of other judges' average probability that the given team wins in Round of 64. Specifically, the jth element is a vector of the judges' predictions of the other judges' average probabilities in the jth game.

`P_NCAA_R16` is a list of the judges' predictions of other judges' average probability that the given team wins in Round of 16. Specifically, the jth element is a vector of the judges' predictions of the other judges' average probabilities in the jth game.

`THETA_NCAA_R64` is a vector of the actual outcomes of the games in the Round of 64. Specifically, the jth element is the actual outcome of jth game in Round of 64.

`THETA_NCAA_R16` is a vector of the actual outcomes of the games in the Round of 16. Specifically, the jth element is the actual outcome of jth game in Round of 16.

`ID_NCAA_R64` is a list of the judges' identification numbers in the judgment tasks associated with the Round of 64 games. Specifically, the jth element is a vector of identification numbers of judges' who participated in estimating the probability of a given team winning the jth game of Round of 64. These values make it possible to track a judge across judgment tasks.

`ID_NCAA_R16` is a list of the judges' identification numbers in the judgment tasks associated with the Round of 16 games. Specifically, the jth element is a vector of identification numbers of judges' who participated in estimating the probability of a given team winning the jth game of Round of 16. These values make it possible to track a judge across judgment tasks.

Remark. The elements of each list correspond to the same game. Specifically, the jth elements of `THETA_NCAA_R16`, `E_NCAA_R16`, `P_NCAA_R16`, and `ID_NCAA_R16` represent the true outcome, estimates, the predictions of others, and identification numbers associated with the jth game in the Round of 16.

Source

Asa Palley and Jack Soll. "Extracting the Wisdom of Crowds When Information Is Shared." doi: 10.1287/mnsc.2018.3047

Index