Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

aggreCAT: an R Package for Mathematically Aggregating Expert judgements

Elliot Gould[*]
University of Melbourne

Charles Gray
TBD

Aaron Willcox
University of Melbourne

Rose O'Dea
University of New South Wales

Rebecca Groenewegen
University of Melbourne

David P. Wilkinson
University of Melbourne

Abstract

Structured elicitation protocols, such as the IDEA protocol, may be used to elicit expert judgements in the form of subjective probabilities from multiple experts. Judgements from individual experts about a particular phenomena must therefore be mathematically aggregated into a single prediction. The process of aggregation may be complicated when judgements are elicited with uncertainty bounds, and also when there are several rounds of elicitation. This paper presents the new R package **aggreCAT**, which provides 28 unique aggregation methods for combining individual judgements into a single, probabilistic measure. The aggregation methods were developed as a part of the Defense Advanced Research Projects Agency (DARPA) 'Systematizing Confidence in Open Research and Evidence' (SCORE) programme, which aims to generate confidence scores or estimates of 'claim credibility' for 3000 research claims from the social and behavioural sciences. We provide several worked examples illustrating the underlying mechanics of the aggregation methods. We also describe a general workflow for using the software in practice to facilitate uptake of this software for appropriate use-cases.

Keywords: mathematical aggregation, expert judgement, DARPA SCORE, replicability, R.

1. Introduction

Expert judgement is frequently used to inform forecasting about uncertain future events across a range of disciplines, including ecology, conservation science, human geography, political science, and management (Sutherland 2018). Judgements from groups of experts tend to perform better than a single expert (Goossens 2008), and it is best-practice to elicit judgements from diverse groups so that group members can bring "different perspectives, cross-examine each others' reasoning, and share information", however judgements or forecasts must then be distilled into a single forecast, ideally accompanied by estimates of uncertainty around those estimates (Hanea, Wilkinson, McBride, Lyon, van Ravenzwaaij, Singleton Thorn, Gray, Mandel, Willcox, Gould, and et al. 2021). Judgements from multiple experts may be combined into a single forecast using either behavioural approaches that force experts into forming consensus, or by using mathematical approaches (Goossens 2008).

Although there are a variety of methods for mathematically aggregating expert judgements into single point-predictions, there are few open-source software implementations available to analysts or researchers. The R R Core Team (2017) package **expert**provides three models of expert opinion to combine judgements elicited from groups of experts (CITE) , and **SHELF** implements only a single method (weighted linear pool) for aggregating expert judgements (CITE). Other R packages providing methods to mathematically aggregate expert judgements do so for non-point predictions, for example, **opera**, which generates time-series predictions (CITE). In this paper we present the **aggreCAT** package, which provides 28 different methods for mathematically aggregating judgements within groups of experts into a single forecast.

1.1. DARPA SCORE program and the repliCATS project

The **aggreCAT** package, and the mathematical aggregators therein, were developed by the repliCATS (Collaborative Assessment for Trustworthy Science) project as a part of the SCORE program (Systematizing Confidence in Open Research and Evidence), funded by DARPA (Defense Advanced Research Projects Agency) (Alipourfard, Arendt, Benjamin, Benkler, Bishop, Burstein, Bush, Caverlee, Chen, Clark, and et al. 2021). The SCORE program is the largest replication project in science to date, and aims to build automated tools that can rapidly and reliably assign "Confidence Scores" to research claims from empirical studies in the Social and Behavioural Sciences (SBS). Confidence Scores are quantitative measures of the likely reproducibility or replicability of a research claim or result, and may be used by consumers of Social and Behavioural Sciences research as a proxy measure for their credibility in the absence of replication effort.

Replications are time-consuming and costly (Isager 2020), and studies have shown that replication outcomes can be reliably elicited from researchers (Gordon 2020). Consequently, the DARPA SCORE program generates Confidence Scores using expert elicitation based on two very different strategies – prediction markets (Gordon 2020) and the IDEA protocol (Hemming, Burgman, Hanea, McBride, and Wintle 2017), the latter of which is used by the repliCATS project (Fraser, Bush, Wintle, Mody, Smith, Hanea, Gould, Hemming, Hamilton, Rumpff, and et al. 2021). **X** of these research claims were randomly selected for direct replication, against which the elicited Confidence Scores are 'ground-truthed'. These findings will aid the development of artificial intelligence tools that can automatically assign Confidence Scores.

The repliCATS IDEA protocol

The repliCATS project adapted and deployed the IDEA protocol to elicit crowd-sourced judgements from diverse groups about the likely replicability of SBS research claims (Fraser *et al.* 2021). The IDEA ('Investigate', 'Discuss', 'Estimate' and 'Aggregate') protocol is a four-step structured elicitation protocol that draws on the 'wisdom of crowds' to elicit subjective judgements about the likelihood of uncertain events (Hemming *et al.* 2017, figure 1). To collect expert judgements about the replicability of SBS claims, we asked participants to estimate the "probability that direct replications of a study would find a statistically significant effect in the same direction as the original claim", eliciting estimates of uncertainty in the form of upper and lower bounds on those point-estimates. Judgements were elicited using the repliCATS platform (Pea 2021), a multi-user cloud-based software platform that implements the IDEA protocol, between July 7th 2019 and November 30th 2020.

For a single claim under assessment, between 4 and 15 experts individually drew on background information to provide estimates of the probability, including 4 numeric data points and one character data point: an upper and lower bound, and best estimate of the event probability, as well as justifications for their estimates, and a value on the likert binary scale up to 7 rating the individuals' degree of comprehension of the claim (Round 1, *Investigate*). In the *Discuss* phase, three-point estimates from each group member are anonymously presented to the group, who then collectively discuss differences in opinion and provide potential evidence for these differences. Group members subsequently provide a second set of probabilistic judgements (Round 2, *Estimate*). Thus, for a single assessment, 2 sets of judgements are elicited from each expert (*pre*- and *post*-group discussion).

During the fourth step, *Aggregate*, judgements are mathematically aggregated into a single *Confidence Score* or forecast of replicability. The repliCATS project developed 28 different methods for mathematically aggregating judgements elicited from groups of experts into Confidence Scores (Hanea *et al.* 2021). We developed the **aggreCAT** package to implement these aggregation methods and deliver Confidence Score for over 3000 SBS research claims for phase one and **X** SBS claims for phase two of the the DARPA SCORE project.

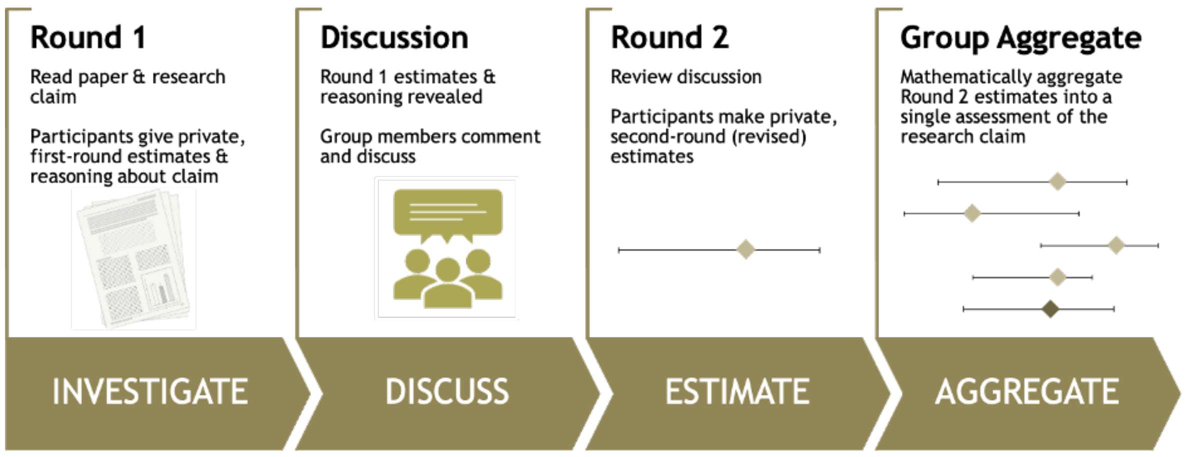

Round 1
Read paper & research claim
Participants give private, first-round estimates & reasoning about claim
Discussion
Round 1 estimates & reasoning revealed
Group members comment and discuss
Round 2
Review discussion
Participants make private, second-round (revised) estimates
Group Aggregate
Mathematically aggregate Round 2 estimates into a single assessment of the research claim
INVESTIGATE
DISCUSS
ESTIMATE
AGGREGATE

Figure 1: The IDEA protocol as deployed by the repliCATS project (reproduced with permission from Wintle et al. 2021).

2. Introducing the aggreCAT package

In this paper we aim to provide a detailed overview of the **aggreCAT** package so that researchers may apply the aggregation functions described in (Hanea *et al.* 2021) to their own expert elicitation datasets where mathematical aggregation is required. Note that judgements elicited using Delphi and other similar elicitation methods that use behavioural or consensus aggregation may not be mathematically aggregated, and thus the **aggreCAT** package is not applicable to datasets collected using such elicitation methods.

We begin by formulating the problem of mathematically aggregating expert judgements. Each method, and its data requirements is summarised in table X (cross-ref). We also briefly summarise package datasets, which were collected by the repliCATS project. By first describing the datasets before describing the aggregation methods in detail, we aim to provide a grounded understanding of the different outputs of expert elicitation using the repliCATS IDEA protocol, and the inputs available to the aggregation functions.

Next, we describe and illustrate the main types of aggregators, which may be categorised according to their data requirements, mathematical properties and computational implementation (SECTION X). By selecting representative functions of each key aggregator type and applying them to a subset of focal claims, we demonstrate the internal mechanics of how these methods differently operationalise the data to generate forecasts or Confidence Scores. We do not give advice on the circumstances in which each method should be used, instead, choice of aggregation method should be informed by the mathematical properties of the method, the desired properties of an aggregation, and the purpose for which the aggregation is being used. For a detailed description of each method as well as a discussion of their relative merits, see (Hanea *et al.* 2021).

Finally, we provide a detailed workflow for aggregating expert judgments for multiple forecasts, using multiple aggregation functions, as implemented by the repliCATS project in the course of delivering 3000 Confidence Scores for the DARPA SCORE program. The **aggreCAT** package provides a set of supporting functions for evaluating or ground-truthing aggregated forecasts or Confidence Scores against a set of known-outcomes, as well as functions for visualising comparisons of different aggregation methods and the outcomes of performance evaluation. We describe this functionality and demonstrate this in the presentation of the repliCATS workflow. The workflow is representative of the probable challenges faced by the researcher in the course of mathematically aggregating groups of forecasts, and should equip the reader to use **aggreCAT** for their own datasets; it exemplifies how to extend the **aggreCAT** package to any expert judgement dataset from any domain in which there are multiple judgements from multiple individuals that need to be combined into a single forecast.

3. Mathematically Aggregating Expert Judgements

Mathematically, the aggregation methods can be divided into three main types:

- Un-weighted linear combination of best estimates, transformed best estimates or distributions,

- Weighted linear combinations of best estimates, transformed best estimates and of distributions, where weights are proxies of forecasting performance constructed from characteristics of participants and/or their judgements, and

- Bayesian methods that use participant judgements as data with which to update both uninformative and informative priors.

However, the **aggreCAT** package user might wish to categorise the aggregation methods according to aspects of their computational implementation and data requirements, because these inform the arguments and the type and form of the data that is parsed to the aggregation functions. These aspects include:

- Elicitation Method, number of elicitation rounds: the majority of aggregation methods require data from only a single round of judgements, i.e. the final post-discussion estimates. However, some aggregation methods require data from both rounds of judgements, which may be elicited using the IDEA protocol or other similar structured elicitation protocol in which there are two rounds of judgements.
- Elicitation method, single point or three point elicitation: several aggregation methods use only a single data point elicited from individuals (their best estimate), however, most aggregation methods require a best estimate, and estimates of uncertainty in the form of upper and lower bounds.
- Number of claims / forecasts assessed by the individual: some weighted aggregation methods consist of weights that are calculated from properties of participant judgements across multiple forecasting questions, not just the target claim being aggregation.
- Supplementary data requirements: several aggregation methods require supplementary data collected either in addition to or as part of the repliCATS IDEA protocol, but which need additional qualitative coding.

The data and structured elicitation protocol requirements are described in the table below (Table 1). All aggregation methods requiring a single round of estimates can therefore be applied to expert judgments derived from any structured elicitation protocol that generates, lower, upper, and best estimates from each individual (i.e. not just the IDEA protocol), and does not enforce behavioural consensus.

Notation and Problem Formulation

Here we describe some preliminary mathematical notation used to represent each aggregation method. For the mathematical specification of each individual aggregation function, please consult (Hanea *et al.* 2021) or the **aggreCAT** package function documentation.

The total number of research claims, *claim*, or unique forecasts being assessed, C , is indexed by $c = 1, ..., C$. The total number of individuals / experts / participants is denoted by N, and is indexed by $i = 1, ..., N$. Each claim assumes binary values, where the value is 0 if the claim is false, and 1 if the claim is true. '`TRUE`' claims are claims where the replication study found a significant result in the same direction as the original research claim, and '`FALSE`' claims are those where the replication study *did not* find a significant result in the same direction as the original study. For each claim c, an individual i assesses the claim as being true or false through providing three probabilities: a lower bound $L_{i,c}$, an upper bound $U_{i,c}$, and a best estimate $B_{i,c}$, satisfying the inequalities: $0 \leq Li, c \leq Bi, c \leq Ui, c \leq 1$.

Every claim is assessed by multiple individuals, and their probabilities are aggregated using one of the 28 aggregation methods to obtain a group or aggregate probability, denoted by $\hat{p}_c$. The aggregated probability calculated using a specific method, is given by $\hat{p}_c\,(MethodID)$. Each aggregation is assigned a unique $MethodID$ which is the abbreviation of the mathematical operation used in calculating the weights. Note that all Best, Lower and Upper estimates are taken to be `round 2` judgements from the repliCATS IDEA protocol Figure 1), unless appended by a "1", where they are `round 1` judgements, e.g. $B1_{i,c}$ denotes the `round 1` Best estimate from individual i for claim c.

Weighting Expert Forecasting Performance

Equal-weighting of judgements are less calibrated, accurate and informative than weighted aggregation methods where judgements from experts who performed well in similar judgement tasks are more heavily weighted (Hanea *et al.* 2021). Proxies for forecasting performance, such as breadth and variability of qualitative reasons used by experts to justify their judgements, can be used to form weights in the absence of measures of experts' prior performance (Hanea *et al.* 2021).

The aggregation methods other than the mean, median and Bayesian approaches in **aggreCAT** each employ weighting schemes that are informed by proxies for good forecasting performance whereby experts' estimates are weighted differently by measures of reasoning, engagement, openness to changing their mind in light of new facts, evidence or opinions presented in the discussion round, extremity of estimates, informativeness of estimates, asymmetry of estimate bounds, granularity of estimates, and by prior statistical knowledge as measured in a quiz.

Below, we define standardised notation for describing weighted linear combinations of individual judgements where un-normalised weights are denoted by w_method and normalised weights by $\tilde{w}\_method$ (Equation 1). All weights must sum to one (be normalised), and that process is the same for all aggregation methods, thus the equations for the aggregation measures are presented for un-normalised weights.

$$\hat{p}_c\,(MethodID) = \frac{1}{N}\sum_{i=1}^{N} \tilde{w}\_method_{i,c} B_{i,c} \tag{1}$$

By default, weights are calculated across all claims on a per-individual, per-claim basis, such that judgements for the same individual are weighted differently across all claims they have provided judgements for. There are some exceptions to this default: `GranWAgg()`, `QuizWAgg()`, `IndIntWAgg()` `IndIntAsymWAgg()`, `VarIndIntWAgg`, `KitchSinkWAgg()`. Note that `IndIntWAgg(),` and methods that include its weighting function as a component, rescale weights by a fixed measure across all claims. Hence, for aggregation methods that use information from multiple claims other than the target claim for which the Confidence Score is being computed, each individual claim c is indexed by $d = 1, ..., C$. Where the default weighting is used, this is coded into each function. However, where more complex and function-specific weighting methods are used, modularised functions have been created for ease of debugging. These function names are prefixed with `weight_`.

3.1. Package datasets

The **aggreCAT** package ships with a core dataset `data_ratings` consisting of judgements elicited during a pilot experiment exploring the performance of IDEA groups in assessing replicability of a set of claims with "known outcomes." "Known-outcome" claims are SBS research claims that have been subject to replication studies in previous large-scale replication projects[1]. Data were collected using the repliCATS IDEA protocol at a two day workshop[2] in the Netherlands, in July 2019, at which 25 participants assessed the replicability of 25 unique SBS claims. In addition to the probabilistic estimates provided for each research claim assessed, participants were also asked to rate the claim's plausibility and comprehensibility, answer whether they were involved in any aspect of the original study, and to provide their reasoning in support of their quantitative estimates, which were used to form measures of reasoning breadth and engagement (Fraser *et al.* 2021).

`data_ratings` is a *tidy* dataframe wherein each *observation* (or row) corresponds to a single value in the set of `values` constituting a participant's complete assessment of a research claim. Each research claim is assigned a unique `paper_id`, and each participant has a unique (and anonymous) `user_name`. The variable `round` denotes the round in which each `value` was elicited (`round_1` or `round_2`). `question` denotes the type of question the `value` pertains to; `direct_replication` for probabilistic judgements about the replicability of the claim, `belief_binary` for participants' belief in the plausibility of the claim, `comprehension` for participants' comprehensibility ratings, and `involved_binary` for involvement in the original study. An additional column `element` maintains the tidy structure of the data, while capturing the multiple `values` that comprise a full assessment of the replicability (`direct_replication`) of a claim; `three_point_best`, `three_point_lower` and `three_point_upper` denote the best estimate and lower and upper bounds respectively. `binary_question` describes the `element` for both the plausibility rating (`belief_binary`)

[1]Many labs 1, 2 and 3 Klein (2014), Klein, Vianello, Hasselman, Adams, Adams, Alper, Aveyard, Axt, Babalola, Bahník, Batra, Berkics, Bernstein, Berry, Bialobrzeska, Binan, Bocian, Brandt, Busching, Redei, Cai, Cambier, Cantarero, Carmichael, Céric, Chandler, Chang, Chatard, Chen, Cheong, Cicero, Coen, Coleman, Collisson, Conway, Corker, Curran, Cushman, Dagona, Dalgar, Rosa, Davis, Bruijn, Schutter, Devos, Vries, Doğulu, Dozo, Dukes, Dunham, Durrheim, Ebersole, Edlund, Eller, English, Finck, Frankowska, Freyre, Friedman, Galliani, Gandi, Ghoshal, Giessner, Gill, Gnambs, Gómez, Gonzalez, Graham, Grahe, Grahek, Green, Hai, Haigh, Haines, Hall, Heffernan, Hicks, Houdek, Huntsinger, Huynh, Ijzerman, Inbar, Innes-Ker, Jimenez-Leal, John, Joy-Gaba, Kamiloğlu, Kappes, Karabati, Karick, Keller, Kende, Kervyn, Knežević, Kovacs, Krueger, Kurapov, Kurtz, Lakens, Lazarević, Levitan, Lewis, Lins, Lipsey, Losee, Maassen, Maitner, Malingumu, Mallett, Marotta, Međedović, Mena-Pacheco, Milfont, Morris, Murphy, Myachykov, Neave, Neijenhuijs, Nelson, Neto, Nichols, Ocampo, O'Donnell, Oikawa, Oikawa, Ong, Orosz, Osowiecka, Packard, Pérez-Sánchez, Petrović, Pilati, Pinter, Podesta, Pogge, Pollmann, Rutchick, Saavedra, Saeri, Salomon, Schmidt, Schönbrodt, Sekerdej, Sirlopú, Skorinko, Smith, Smith-Castro, Smolders, Sobkow, Sowden, Spachtholz, Srivastava, Steiner, Stouten, Street, Sundfelt, Szeto, Szumowska, Tang, Tanzer, Tear, Theriault, Thomae, Torres, Traczyk, Tybur, Ujhelyi, Aert, Assen, van der Hulst, Lange, Veer, Echeverría, Vaughn, Vázquez, Vega, Verniers, Verschoor, Voermans, Vranka, Welch, Wichman, Williams, Wood, Woodzicka, Wronska, Young, Zelenski, Zhi-jia, and Nosek (2018), Ebersole (2016), the Social Sciences Replication Project Camerer (2018) and the Reproducibility Project Psychology aac (2015).

[2]See Hanea *et al.* (2021) for details. The workshop was held at the annual meeting of the Society for the Improvement of Psychological Science (SIPS), <https://osf.io/ndzpt/>.

and involvement (`involved_binary`) questions, whereas `likert_binary` is the `element` describing a participant's `comprehension` rating. judgements are recorded in column `value` in the form of percentage probabilities ranging from (0,100). The `binary_questions` corresponding to comprehensibility and involvement consist of binary values (`1` for the affirmative, and `-1` for the negative). Finally, values corresponding to participants' comprehension ratings are on a `likert_binary` scale from `1` through `7`. Below we show some example data for a single user for a single claim to illustrate this structure of the core `data_ratings` dataset.

```
R> library(tidyverse,quietly = TRUE)
R> library(aggreCAT)
R> data(data_ratings)
R> data_ratings %>%
+   print(n = 18)
```

```
# A tibble: 6,880 x 7
   round   paper_id user_name  question            element  value group
   <chr>   <chr>    <chr>      <chr>               <chr>    <dbl> <chr>
 1 round_1 100      fx3d4tmdhh direct_replication three_p~    30 UOM1
 2 round_1 100      fx3d4tmdhh involved_binary    binary_~    -1 UOM1
 3 round_1 100      fx3d4tmdhh belief_binary      binary_~    -1 UOM1
 4 round_1 100      fx3d4tmdhh direct_replication three_p~    40 UOM1
 5 round_1 100      fx3d4tmdhh direct_replication three_p~    45 UOM1
 6 round_1 100      fx3d4tmdhh comprehension      likert_~     5 UOM1
 7 round_1 100      sv2yl8jszy direct_replication three_p~    60 UOM1
 8 round_1 100      sv2yl8jszy direct_replication three_p~    90 UOM1
 9 round_1 100      sv2yl8jszy direct_replication three_p~    75 UOM1
10 round_1 100      sv2yl8jszy comprehension      likert_~     7 UOM1
11 round_1 100      sv2yl8jszy involved_binary    binary_~    -1 UOM1
12 round_1 100      sv2yl8jszy belief_binary      binary_~     1 UOM1
13 round_1 100      v6n605nzv1 direct_replication three_p~    40 UOM1
14 round_1 100      v6n605nzv1 comprehension      likert_~     5 UOM1
15 round_1 100      v6n605nzv1 belief_binary      binary_~     1 UOM1
16 round_1 100      v6n605nzv1 direct_replication three_p~    80 UOM1
17 round_1 100      v6n605nzv1 direct_replication three_p~    65 UOM1
18 round_1 100      v6n605nzv1 involved_binary    binary_~    -1 UOM1
# ... with 6,862 more rows
```

Not all data necessary for constructing weights on performance is contained in `data_ratings`. Additional data collected as part of the repliCATS IDEA protocol are contained within separate datasets to `data_ratings`. Justifications for giving particular judgements are contained in `data_justifications`. on the repliCATS platform users were given the option to comment on others' justifications (`data_comments`), to vote on others' comments (`data_comment_ratings`) and on others' justifications (`data_justification_ratings`). Finally, **aggreCAT** contains three 'supplementary' datasets containing data collected externally to the repliCATS IDEA protocol: `data_supp_quiz`, `data_supp_priors`, and `data_supp_reasons`.

Quiz Score Data

Prior to the workshop, participants also completed an optional quiz on statistical concepts and meta-research that we expect participants to be aware of in order to reliably evaluate the replicability of research claims. Quiz responses are contained in `data_supp_quiz` and are used to construct performance weights for the aggregation method `QuizWAgg` where each participant receives a `quiz_score` from 0 - **X (TODO)** if they completed the quiz, and `NA` if they did not attempt or fully complete the quiz (see Hanea *et al.* 2021, for further details). (Question for Bonnie, possibly Rose?: Pretty sure they get points for any question they completed, even if they didn't finish)

Reasoning Data

`ReasonWAgg` uses the number of unique reasons given by participants to support a Best Estimate for a given claim $B_{i,c}$ to construct performance weights, and is contained within `data_supp_reasons`. Qualitative statements made by individuals during claim evaluation were recorded on the repliCATS platform (Pea 2021) and coded as falling into one of 25 unique reasoning categories by the repliCATS Reasoning team (Wintle 2021). Reasoning categories include plausibility of the claim, effect size, sample size, presence of a power analysis, transparency of reporting, and journal reporting (Hanea *et al.* 2021). Within `data_supp_reasons`, each of the 25 categories of reasoning are distributed as columns in the dataset, and for each claim `paper_id`, each participant `user_id` is assigned a logical `1` or `0` if they included that reasoning category in support of their Best estimate for that claim. See section ref(ReasonWAgg) for details on the `ReasonWAgg` aggregation method.

Bayesian Prior Data

`BayPRIORsAgg()` uses Bayesian updating to update a prior probability of a claim replicating estimated from a predictive model (Gould, Willcox, Fraser, Singleton Thorn, and Wilkinson 2021) using an aggregate of the best estimates for all participants assessing a given claim c (Hanea *et al.* 2021). The prior data is contained in `data_supp_priors` with each claim in column `paper_id` being assigned a prior probability of the claim replicating (on the logit scale) in column `prior_means`. (**TODO** should explain further about the mean / median of the distribution, ie internal workings of BayPRIORsAgg??).

Aggregation Wrapper Functions

Although there are **n** aggregation methods in total, we grouped methods based on their mathematical properties into eight 'wrapper' functions, denoted by the suffix `WAgg`, the abbreviation of *weighted aggregation*: `LinearWAgg()`, `AverageWAgg()`, `BayesianWAgg()`, `IntervalWAgg()`, `ShiftingWAgg()`, `ReasoningWAgg()`, `DistributionWAgg()`, and `ExtremisationWAgg()`. The specific aggregation method is applied according to the `type` argument, whose options are described in each aggregation wrapper functions' help page.

3.2. 'Tidy' Aggregation and Prescribed Inputs

The design philosophy of **aggreCAT** is principled on 'tidy' data (Wickham 2014). Each aggregation method takes a '`tibble`' of judgements (`data_ratings`) as its input, and returns a '`tibble`' consisting of the variables `method`, `paper_id`, `cs` and `n_experts` (see section ref(ArMean) for illustration of outputs); where `method` is a character vector corresponding to the aggregation method name. Each aggregation is applied as a summary function (Wickham and Grolemund 2017b), and therefore returns a single row or observation containing a single confidence score `cs` for each claim or `paper_id`. The number of expert judgements aggregated in the confidence score is returned in the column `n_experts`. Because of the tidy nature of the aggregation outputs, multiple aggregations can be applied to the same data with the results of all aggregation methods bound together in a single dataframe.

Each aggregation function requires values derived from three-point elicitation (best-estimate, upper and lower bound).

For every aggregation function, the three-point elicitation values corresponding to the `"direct_replication"` question are required inputs. Of the question and elements other than the three-point elicitation elements belonging to the direct replication question, only the `comprehension` question with the `likert_binary` elements is required – this is an input into `aggreCAT::CompWAgg`, which is used to weight participants judgements. Each value provided by a participant is timestamped, but this is not a required data field.

4. Focal Claim Aggregation

We now demonstrate how judgements elicited from a diverse group of individuals may be mathematically aggregated for a single forecasting problem, using the datasets packaged with **aggreCAT**. We demonstrate the internal mechanics of the weighting methods and the different data requirements of each of the different types of aggregators – namely; methods with non-weighted linear combinations of judgements, weighted linear combinations of judgements, re-scaled weighted linear combinations of judgements, methods that require supplementary data, and methods that require data elicited from the full IDEA protocol. Each group of methods differs in the type of judgements elicited (single point- or three-point estimates), the number of elicitation rounds (one or two rounds), whether multiple forecasts / elicited judgements are used during confidence score computation for a target forecast / claim, and finally whether supplementary data is required for aggregation.

Here we demonstrate the application of aggregation methods for each group of methods using set of 'focal claims' selected from the pilot study dataset supplied with the **aggreCAT** package. Below we subset the dataset `data_ratings` to include a sample of five claims with judgements from five randomly-sampled participants. From these focal claims, we select a target claim `czttvy` for which we will apply an exemplar aggregation method from each mathematical aggregator (Table 1).

```
R> set.seed(1234)
R> focal_claims <- data_ratings %>%
+  filter(paper_id %in% c("24", "138", "186", "108"))
R> # select 5 users to highlight in focal claim demonstration
R> focal_users <- focal_claims %>%
+  distinct(user_name) %>%
+   sample_n(5) %>%
+   mutate(participant_name = paste("participant", rep(1:n())))
R> # filter out non-focal users from focal claims
R> focal_claims <- focal_claims %>%
+   right_join(focal_users, by = "user_name") %>%
+   select(-user_name) %>%
+   rename(user_name = participant_name)
R> focal_claims

# A tibble: 220 x 7
   round   paper_id question           element     value group user_~1
   <chr>   <chr>    <chr>              <chr>       <dbl> <chr> <chr>
 1 round_1 108      comprehension      likert_bin~     7 UOM1  partic~
 2 round_1 108      direct_replication three_poin~    90 UOM1  partic~
 3 round_1 108      direct_replication three_poin~    40 UOM1  partic~
 4 round_1 108      belief_binary      binary_que~     1 UOM1  partic~
 5 round_1 108      involved_binary    binary_que~    -1 UOM1  partic~
 6 round_1 108      direct_replication three_poin~    65 UOM1  partic~
 7 round_1 108      direct_replication three_poin~    60 UOM3  partic~
 8 round_1 108      direct_replication three_poin~    40 UOM3  partic~
 9 round_1 108      direct_replication three_poin~    51 UOM3  partic~
10 round_1 108      comprehension      likert_bin~     6 UOM3  partic~
# ... with 210 more rows, and abbreviated variable name 1: user_name
```

Claim ID	User Name	Lower Bound	Best Estimate	Upper Bound
108	participant 1	70	85	90
108	participant 2	70	80	90
108	participant 3	40	65	90
108	participant 4	60	80	90
108	participant 5	50	60	70

Table 1: Focal Claim Data: expert judgements for claim czttvy derived from a subset of 5 claims and 5 participants from data_ratings. Judgements are displayed as percentages.

4.1. Non-weighted linear combination of judgements

We first demonstrate the mechanics of mathematical aggregation and its implementation using the **aggreCAT** package with the simplest, unweighted aggregation method, `ArMean`. All other aggregation methods take this underlying computational blueprint, and expand on it according to the aggregation methods' requirements (See Box 1 for details). `ArMean` (Equation 2) takes the unweighted linear average (i.e. arithmetic mean) of the best estimates, $B_{i,c}$.

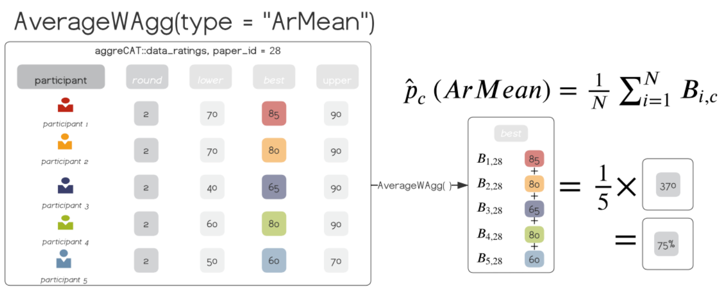
$$\hat{p}_c\left(ArMean\right) = \frac{1}{N}\sum_{i=1}^{N} B_{i,c} \tag{2}$$

Below we demonstrate the application of `ArMean` on a single claim `czttvy` for a subset of participants who assessed this claim. We also illustrate this aggregation visually in Figure 2. `ArMean` is applied using the aggregation method `AverageWAgg()`, which is a wrapper function for several aggregation methods that calculate different types of averaged best-estimates (`?AverageWAgg`). The function returns the Confidence Score for the claim in the form of a '`tibble`':

```
R> focal_claims %>%
+  filter(paper_id == "108") %>%
+  AverageWAgg(type = "ArMean")
```

```
-- AverageWAgg: ArMean ----------------------------------------------



-- Pre-Processing Options --



i Round Filter: TRUE

i Three Point Filter: TRUE

i Percent Toggle: FALSE

# A tibble: 1 x 4
  method paper_id    cs n_experts
  <chr>  <chr>    <dbl>     <int>
1 ArMean 108         74         5
```

- wrapper functions and the 'type' argument
- default arg structure for each wrapper fun
  - percent toggle
  - placeholder
  - name
  - What else?
- COmputation:


AverageWAgg(type = "ArMean")
aggreCAT::data_ratings, paper_id = 28
participant
round
lower
best
upper
participant 1
participant 2
participant 3
participant 4
participant 5
AverageWAgg( )
$\hat{p}_c\,(ArMean) = \frac{1}{N}\sum_{i=1}^{N} B_{i,c}$
$B_{1,28}$ 85 + $B_{2,28}$ 80 + $B_{3,28}$ 65 + $B_{4,28}$ 80 + $B_{5,28}$ 60
$= \frac{1}{5} \times$ 370
= 75%

participant	round	lower	best	upper
participant 1	2	70	85	90
participant 2	2	70	80	90
participant 3	2	40	65	90
participant 4	2	60	80	90
participant 5	2	50	60	70

Figure 2: `ArMean()` uses the Estimates (shown in colour) from each participant to compute the mean. We illustrate this using a single claim `zttvyg` for a subset of 5 out of 25 participants from the `data_ratings` dataset. Note that the data representations in this figure are for explanatory purposes only, the data in the actual aggregation is tidy, with long form structure and format.

- Weighting Functions
- Note that not all functions use a separately defined weighting function – for simpler weight computations, these are defined in-function rather than being modularised See Table 1.

Each aggregation function follow a general workflow whereby the primary dataset `data_ratings`, parsed to the `expert_judgements` argument, is first pre-processed by `pre_process_judgements()`, subsequently the aggregation method is applied using `dplyr::summarise()`, and then finally the aggregated data is parsed to `postprocess_judgements()`. This general aggregation method workflow is seen best in `ArMean()`:

```
AverageWAgg

function (expert_judgements, type = "ArMean", name = NULL, placeholder = FALSE,
    percent_toggle = FALSE)
{
    if (!(type %in% c("ArMean", "GeoMean", "Median", "LOArMean",
        "LOGeoMean", "ProbitArMean"))) {
        stop("`type` must be one of \"ArMean\", \"GeoMean\", \"Median\", \"LOArMean\", \
    }
    name <- ifelse(is.null(name), type, name)
    cli::cli_h1(sprintf("AverageWAgg: %s", name))
    if (isTRUE(placeholder)) {
        method_placeholder(expert_judgements, name)
    }
    else {
        df <- expert_judgements %>% preprocess_judgements(percent_toggle = {
            {
                percent_toggle
            }
        }) %>% dplyr::filter(element == "three_point_best") %>%
            dplyr::group_by(paper_id)
        switch(type, ArMean = {
            df <- df %>% dplyr::summarise(aggregated_judgement = mean(value,
                na.rm = TRUE), n_experts = dplyr::n())
        }, GeoMean = {
            df <- df %>% dplyr::summarise(n_experts = dplyr::n(),
                aggregated_judgement = (prod(value, na.rm = TRUE))^(1/n_experts))
        }, Median = {
            df <- df %>% dplyr::summarise(aggregated_judgement = median(value,
                na.rm = TRUE), n_experts = dplyr::n())
        }, LOArMean = {
            if (any(df$value < 0) | any(df$value > 1)) {
                stop("LOArMean requires probabilistic judgements. Check your data compat
            }
            df <- df %>% dplyr::mutate(value = dplyr::case_when(value ==
                1 ~ value - .Machine$double.eps, value == 0 ~
                value + .Machine$double.eps, TRUE ~ value), log_odds = log(abs(value/(1
                value)))) %>% dplyr::summarise(aggregated_judgement = mean(log_odds,
                na.rm = TRUE), n_experts = dplyr::n()) %>% dplyr::mutate(aggregated_judg
                exp(aggregated_judgement)))
        }, LOGeoMean = {
            if (any(df$value < 0) | any(df$value > 1)) {
                stop("LOGeoMean requires probabilistic judgements. Check your data compa
            }
            df <- df %>% dplyr::mutate(value = dplyr::case_when(value ==
                1 ~ value - .Machine$double.eps, value == 0 ~
                value + .Machine$double.eps, value == 0.5 ~ value +
                .Machine$double.eps, TRUE ~ value), log_odds = log(abs(value/(1 -
                value)))) %>% dplyr::summarise(n_experts = dplyr::n(),
                aggregated_judgement = (prod(log_odds, na.rm = TRUE))^(1/n_experts)) %>%
                dplyr::mutate(aggregated_judgement = exp(aggregated_judgement)/(1 +
                  exp(aggregated_judgement)))
        }, ProbitArMean = {
            df <- df %>% dplyr::mutate(probit = VGAM::probitlink(value,
                bvalue = .Machine$double.eps)) %>% dplyr::summarise(aggregated_judgement
                na.rm = TRUE), n_experts = dplyr::n()) %>% dplyr::mutate(aggregated_judg
                inverse = TRUE))
        })
        df %>% dplyr::mutate(method = name) %>% postprocess_judgements()
    }
}
<bytecode: 0x7f8bb0221a70>
<environment: namespace:aggreCAT>
```

The `preprocess_judgements()` function parses the primary dataset `data_ratings` through the argument `expert_judgements` to filter the required quantitative inputs for the aggregation method at hand. It uses two filtering arguments to control which round of judgements are used as inputs (`round_2_filter`), and whether the full set of three-point elicitation judgements should be used, or whether other additional elements must be returned (`three_point_filter`), including the `likert_binary` elements for participants' comprehensibility ratings, and the plausibility ratings under `binary_question` in column `element`. `three_point_filter` defaults to `TRUE` to provide only direct replication questions and associated values. Nearly all aggregation functions use only the round 2 judgements, so the `round_2_filter` defaults to `TRUE` (See Table 1 for required inputs of all aggregation methods). `preprocess_judgements()` further pre-processes the data to remove missing data, and to return the data into an appropriate structure for applying the aggregation function with `dplyr::summarise()`.

```
R> data_ratings %>%
+  group_by(paper_id) %>%
+  nest()  %>%
+  ungroup() %>%
+  sample_n(1) %>%
+  unnest(cols = c(data)) %>%
+  preprocess_judgements()



-- Pre-Processing Options --



i Round Filter: TRUE

i Three Point Filter: TRUE

i Percent Toggle: FALSE

# A tibble: 75 x 5
   round   paper_id user_name  element           value
   <chr>   <chr>    <chr>      <chr>             <dbl>
 1 round_2 118      fx3d4tmdhh three_point_best     50
 2 round_2 118      fx3d4tmdhh three_point_upper    60
 3 round_2 118      fx3d4tmdhh three_point_lower    40
 4 round_2 118      sv2yl8jszy three_point_best     45
 5 round_2 118      sv2yl8jszy three_point_upper    70
 6 round_2 118      sv2yl8jszy three_point_lower    30
 7 round_2 118      v6n605nzv1 three_point_best     50
 8 round_2 118      v6n605nzv1 three_point_lower    40
 9 round_2 118      v6n605nzv1 three_point_upper    60
10 round_2 118      033t8xcqan three_point_best     64
# ... with 65 more rows
```

After `preprocessing_judgements()` and the aggregation method is applied, the function `post_process_judgements()` then processes the variables into the final data frame that is returned by each aggregation function. The post processing function returns a '`tibble`' consisting of observations equal to the number of unique claims that were parsed to `post_process_judgements()`, the `method`, associated `method_id` , `paper_id` , the Confidence Score `cs`, as well as the number of participants `n_experts` whose assessments were used in the aggregation, and the date of the first and last assessments `first_expert_date` and `last_expert_date` respectively.

4.2. Weighted linear combinations of judgements

We now demonstrate the construction of weights for forecasting performance, as well as the use of uncertainty bounds in addition to the Best Estimates (i.e. three-point estimates) in the aggregation computation. The aggregation method `IntWAgg` weights each participant's best estimate $B_{i,c}$ by the width of their uncertainty intervals, i.e. the difference between an individual's upper $U_{i,c}$ and lower bounds $L_{i,c}$. For a given claim c, a vector of weights for all individuals is calculated from their upper and lower estimates using the weighting function, `weight_interval()`, which calculates the interval width for each individual's estimate for the target claim. The weights are then normalised across the claim (by dividing each weight by the sum of all weights per claim). Normalised weights are then multiplied by the corresponding individual's best estimates $B_{i,c}$ andsummed together into a single Confidence Score (Figure 3).

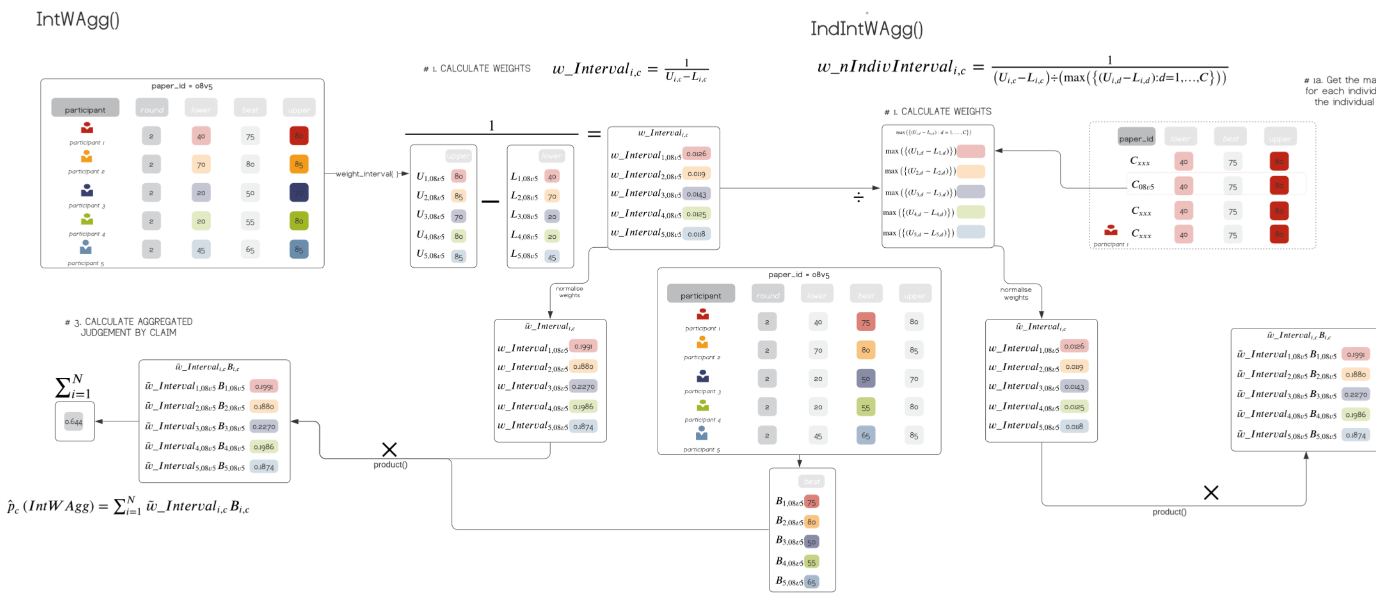
4.3. Re-scaled weighted linear combinations of judgements

Individuals vary in the interval widths they give across different claims. `IndIntWAgg` is a variation on `IntWAgg` that accounts for cross-claim variation within individuals' assessments by rescaling the interval width weights for individual i for claim c relative to the widest interval provided by that individual across all claims C, (Equation 4). For the target claim, each individual's interval width is divided by the maximum interval width that same individual gave across all claims they have provided judgements for, using the weighting function `weight_nIndivInterval()` (Equation 3). The process of re-scaling is illustrated in Figure 3. Other aggregation methods that re-scale weights by using data from multiple claims other than the target claim under aggregation are `VarIndIntWAgg`, `IndIntAsymWAgg`, `KitchSinkWAgg` (applied with the wrapper function `IntervalWAgg()`) and `GranWAgg` (applied with the wrapper function `LinearWAgg()`), see Table 1.

$$w\_Interval_{i,c} = \frac{1}{U_{i,c} - L_{i,c}} \tag{3}$$

$$\hat{p}_c\left(IntWAgg\right) = \sum_{i=1}^{N} \tilde{w}\_Interval_{i,c} B_{i,c} \tag{4}$$

As for `AverageWAgg()`, we supply the aggregation method names as a character vector to the type, but in this instance we do so via the **purrr** function `map_dfr()`, which row-binds



IntWAgg()
IndIntWAgg()
# 1. CALCULATE WEIGHTS
$w\_Interval_{i,c} = \frac{1}{U_{i,c} - L_{i,c}}$
$w\_nIndivInterval_{i,c} = \frac{1}{(U_{i,c}-L_{i,c}) \div (\max(\{(U_{i,d}-L_{i,d}): d=1,\ldots,C\}))}$
paper_id = 08v5
participant
round
lower
best
upper
weight_interval()
normalise weights
# 3. CALCULATE AGGREGATED JUDGEMENT BY CLAIM
product()
0.644
$\hat{p}_c\,(IntWAgg) = \sum_{i=1}^{N} \tilde{w}\_Interval_{i,c}\, B_{i,c}$

Figure 3: Example applications of mathematical aggregation methods a) `IntWAgg` and b) `IndIntWAgg` using the wrapper function a1) `IntWAgg` uses participants' upper and lower bounds to construct performance weights. b2) This weighting computation is modified in `IndIntWAgg` whereby the weights for each individual are re-scaled by the largest interval width across all claims for a given individual. We exemplify this rescaling process by illustrating the calculation of participant 1's maximum interval width across all claims they assessed in the demonstration dataset `focal_claims`. This is repeated for every individual who has assessed the target claim under aggregation.

the results of each application of `IntervalWAgg()` into a single '`tibble`' with the resultant Confidence Scores:

```
R> focal_claims %>%
+  purrr::map_dfr(.x = c("IndIntWAgg", "IntWAgg"),
+                 .f = ~ aggreCAT::IntervalWAgg(expert_judgements = focal_claims %>%
+                                                 dplyr::filter(paper_id == "108"),
+                                               type = .x)
+  )
```

```
-- IntervalWAgg: IndIntWAgg ------------------------------------------

-- Pre-Processing Options --

i Round Filter: TRUE

i Three Point Filter: TRUE

i Percent Toggle: FALSE

-- IntervalWAgg: IntWAgg ---------------------------------------------

-- Pre-Processing Options --

i Round Filter: TRUE

i Three Point Filter: TRUE

i Percent Toggle: FALSE
```

```
# A tibble: 2 x 4
  method      paper_id    cs n_experts
  <chr>       <chr>    <dbl>     <int>
1 IndIntWAgg 108       74            5
2 IntWAgg    108       74.8          5
```

4.4. Aggregation Methods Requiring Supplementary Data

In addition to the three-point elicitation dataset `data_ratings`, Some aggregation methods require supplementary data inputs collected externally to the repliCATS IDEA protocol. Each aggregation wrapper function that requires supplementary data expects this data to be provided as a '`data.frame`' or '`tibble`' in addition to the main judgements that are provided to the expert_judements argument. Aggregation methods requiring supplementary data, include `ReasonWAgg` and `ReasonWAgg2` (applied with `ReasoningWAgg()`), `QuizWAgg` applied with **TODO: what wrapper function??** and `BayPRIORsAgg` (applied with `BayesianWAgg()`). Finally, `EngWAgg` requires data summarised forms of data collected by the repliCATS IDEA protocol, but not contained in `data_ratings`, see Table 1 for details.

We illustrate the usage and internal mechanics of this type of aggregation with the method `ReasonWAgg`, which weights participants' best estimates $B_{i,c}$ by the breadth of reasoning provided to support the individuals' estimate (Equation 5). This method is premised on the expectation that multiple (unique) reasons justifying an individual's judgement may indicate their breadth of thinking, understanding and knowledge about both the claim and its context (Hanea *et al.* 2021) while also reflecting their level of engagement and general conscientiousness. These qualities are correlated with improved forecasting (Wintle 2021). Thus, greater weighting of best estimates that are accompanied by a greater number of supporting reasons may yield more reliable Confidence Scores.

$$\hat{p}_c\left(ReasonWAgg\right) = \sum_{i=1}^{N} \tilde{w}\_reason_{i,c} B_{i,c} \tag{5}$$

`ReasonWAgg` is applied with the wrapper function `ReasoningWAgg()`, which uses the the coded reasoning data `data_supp_reasons` (Section 3.1.2) to compute a vector of weights, $w\_reason_{i,c}$, the number of unique reasons provided by individual i in support of their estimate for claim c (Figure 4). Weights are then normalised across individuals, multiplied by the Best Estimates for that claim $B_{i,c}$ and weighted best estimates are then summed to generate the Confidence Score (Equation 5).

The focal claim selected for aggregation using ReasonWAgg is `09xkh8`, the round 2 three-point estimates from the five focal participants for this claim are shown in Table 2. We first prepare the supplementary data for aggregation `data_supp_reasons`, subsetting only the participants contained in our `focal_claims` dataset. We also illustrate a subset of the supplementary data for our 5 focal participants for the focal claim `09xkh8` (see `?data_supp_reasons` for a description of variables):

```
R> data_supp_reasons_focal <- aggreCAT::data_supp_reasons %>%
```

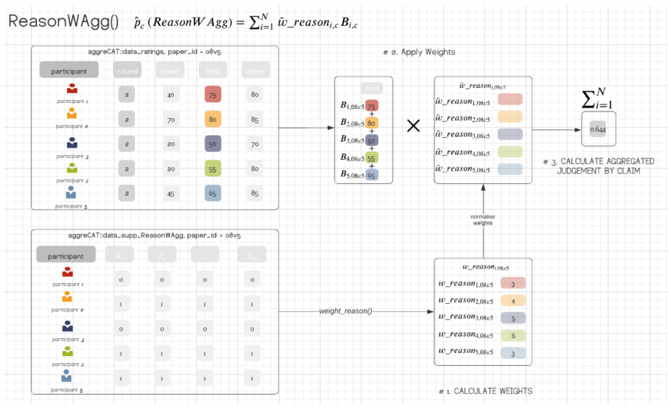


Figure 4: Illustration of the `ReasonWAgg` aggregation method for a subset of five participants who assessed claim `09xkh8`. `ReasonWAgg` is applied using the wrapper function `ReasoningWAgg()` and exemplifies aggregation methods that use supplementary data (`data\_supp\_ReasonWAgg`) collected externally to the IDEA protocol in the construction of weights and subsequent calculation of Confidence Scores. Weights are constructed by taking the sum of the number of unique reasons made in support of quantitative estimates for each participant, for the target claim.

```
+  dplyr::right_join(focal_users) %>%
+  dplyr::select(-user_name) %>%
+  dplyr::rename(user_name = participant_name)

Joining, by = "user_name"

R> data_supp_reasons_focal %>%
+  dplyr::filter( paper_id == 24) %>%
+  tidyr::pivot_longer(cols = c(-paper_id, -user_name)) %>%
+  dplyr::arrange(name) %>%
+  tidyr::separate(name, into = c("reason_num", "reason"), sep = "\\s", extra = "merge") %
+  dplyr::select(-reason) %>%
+  dplyr::group_by(paper_id, user_name) %>%
+  tidyr::pivot_wider(names_from = reason_num) %>%
+  dplyr::arrange(user_name)

# A tibble: 5 x 15
# Groups:   paper_id, user_name [5]
  paper_id user_name    RW05  RW09  RW11  RW12  RW13  RW14  RW15  RW16
  <chr>    <chr>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 24       participan~     0     0     1     0     0     0     1     1
2 24       participan~     0     0     1     0     0     0     2     1
3 24       participan~     0     0     0     0     0     1     0     0
4 24       participan~     0     0     0     0     0     0     0     1
5 24       participan~     0     0     0     0     0     0     0     0
# ... with 5 more variables: RW18 <dbl>, RW19 <dbl>, RW22 <dbl>,
#   RW23 <dbl>, RW24 <dbl>
```

Claim ID	User Name	Lower Bound	Best Estimate	Upper Bound
24	participant 1	5	20	40
24	participant 2	5	11	17
24	participant 3	20	35	50
24	participant 4	10	15	20
24	participant 5	10	30	50

Table 2: Focal Claim 09xkh8 judgements comprising best estimates, upper and lower bounds elicited from 5 participants. Judgements are displayed as percentages.

Confidence Scores estimating the replicability for claim `09xkh8` (Table Table 2) using the `ReasonWAgg` method are computed using `ReasoningWAgg()` and by providing the supplementary data to the reasons argument:

```
R> focal_claims %>%
+  dplyr::filter(paper_id == "24") %>%
+  aggreCAT::ReasoningWAgg(reasons = data_supp_reasons_focal,
+                          type = "ReasonWAgg")
```

4.5. Bayesian Aggregation Methods

Both Bayesian methods `BayTriVar` and `BayPRIORsAgg` use the full three-point elicitation data, i.e., they use information contained in the uncertainty bound provided by individuals (upper $U_{i,c}$ and lower bounds $L_{i,c}$), in addition to Best Estimates, $B_{i,c}$. Like `IndIntWAgg` and other methods (Table 1), the Bayesian aggregation methods also construct weights from information encoded in participant assessments of claims other than the target claim under aggregation. In fact, the Bayesian methods require more than a single claim's worth of data to work properly execute due mathematical specification of the models (See `?BayesianWAgg` and below for details).

The two Bayesian methods use the elicited probabilities as data to update prior probabilities. `BayTriVar` incorporates three sources of uncertainty in best estimates: variability in best estimates across all claims, variability in estimates across all individuals, and claim-participant variability (which is derived from an individuals' upper and lower bounds). This Bayesian model, implemented using **R2JAGS**(Su and Yajima 2020), takes the log odds transformed individual best estimates, and uses a normal likelihood function to derive a posterior distribution for the probability of replication. The estimated confidence score is the mean of this posterior distribution.

`BayPRIORsAgg` is a modified version of `BayTriVar` where, instead of using default priors, priors are generated from a predictive model that estimates the probability of a claim replicating based on characteristics of the claim and publication (Gould *et al.* 2021). Priors are parsed as supplementary data to the wrapper function `BayesianWAgg()` using the argument priors (section Section 3.1.3) with each claim having its own unique prior.

We illustrate aggregation of participant judgements using the method `BayTriVar` to generate a Confidence Score for the claim `czttvy`. Note that `BayesianWAgg()` expects best estimates in the form of probabilities, so to convert elicited values in the form of percentages within the data parsed to expert_judgements to probabilities, the logical value `TRUE` is supplied to the argument percent_toggle:

```
R> focal_claims %>%
+  BayesianWAgg(type = "BayTriVar",
+               percent_toggle = TRUE) %>%
+  dplyr::filter(paper_id == "108")

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 20
   Unobserved stochastic nodes: 4
   Total graph size: 230

Initializing model


# A tibble: 1 x 4
  method    paper_id    cs n_experts
  <chr>     <chr>    <dbl>     <int>
1 BayTriVar 108      0.699         5
```

The Confidence Score calculated for a given claim depends on data for other claims and participants included in the expert_judgements argument other than the target claim, because, by definition, `bayesianWAgg()` calculates the Confidence Score for a target claim using data from participants' assessments of other claims, and from all other claims in the dataframe parsed to the expert_judgements argument. Because information about other claims than the target claim is used to calculate the Confidence Score for the target claim, what is included in the data supplied to the argument expert_judgements in `bayesianWAgg()` will alter the Confidence Score. Above, we calculated the Confidence Score for claim `czttvy` but including information from 3 additional claims included in the `focal_claims` dataframe: 108, 138, 186, 24. However, if we were to supply assessments for only two claims to `BayesianWAGG()`, then we would observe a different result for focal claim `czttvy`:

```
R> focal_claims %>%
+  dplyr::filter(paper_id %in% c("108", "138")) %>%
+  aggreCAT::BayesianWAgg(type = "BayTriVar", percent_toggle = TRUE) %>%
+  dplyr::filter(paper_id == "108")

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 10
   Unobserved stochastic nodes: 2
   Total graph size: 116

Initializing model

# A tibble: 1 x 4
  method    paper_id    cs n_experts
  <chr>     <chr>    <dbl>     <int>
1 BayTriVar 108      0.739         5
```

The Confidence Score shifts from 0.7 to 0.74. Note that `BayesianWAgg()` cannot calculate confidence scores when judgements for only a single claim is provided to `expert_judgements()`, because by definition the underlying Bayesian model calculates variance across multiple claims and multiple participants:

```
R> focal_claims %>%
+   dplyr::filter(paper_id == "108") %>%
+   aggreCAT::BayesianWAgg(type = "BayTriVar", percent_toggle = TRUE)

Error in `aggreCAT::BayesianWAgg()`:
! Model requires n > 1 ids to successfully execute.
```

Finally, all of the previous methods illustrated in this section have been used with data generated using the IDEA elicitation protocol, however this elicitation method is not strictly necessary for the of these methods. Methods that *do* require the full IDEA protocol for their correct mathematical implementation, such as `ShiftingWAgg()`, which use two rounds of three-point judgements in which the second round jdugements are revised after discussion, are listed in Table 1.

5. An illustrative workflow for use in real study contexts

During phase one of the DARPA SCORE program, 509 participants assessed 3000 unique claims using the repliCATS IDEA protocol. This required us to batch aggregation over multiple claims, and to generate Confidence Scores for multiple claims. We also applied multiple aggregation methods to the same claim so that we could compare and evaluate the different aggregation methods. We expect that these are not uncommon use-cases,consequently in this section we demonstrate a general workflow for using the **aggreCAT** package to aggregate expert judgements using pilot data from DARPA SCORE program generated by the repliCATS project.

5.1. Generating multiple forecasts

During expert-elicitation the analyst or researcher may be tasked with generating multiple forecasts for different problems or questions, and therefore it is useful to batch the aggregation. Since the **aggreCAT** package is designed using the principles of *tidy* data analysis (Wickham, Averick, Bryan, Chang, McGowan, François, Grolemund, Hayes, Henry, Hester, Kuhn, Pedersen, Miller, Bache, Müller, Ooms, Robinson, Seidel, Spinu, Takahashi, Vaughan, Wilke, Woo, and Yutani 2019), each aggregation function accepts a dataframe of raw three-point forecasts for one or more claims, C, parsed to the argument `expert_judgements`. The data pre-processing and aggregation methods are applied using a combination of calls to **tidyverse** functions, including `summarise` and `mutate`. From the user's perspective, this means that data processing and application of the aggergation methods is handled internally by the **aggreCAT** package, rather than by the user. The user is therefore free to focus their attention on the interpretation and analysis of the forecasts. Here we demonstrate the application of the `ArMean` aggregation method to four focal claims simultaneously:

```
AverageWAgg(focal_claims, type = "ArMean")
```

```
-- AverageWAgg: ArMean ----------------------------------------------



-- Pre-Processing Options --



i Round Filter: TRUE

i Three Point Filter: TRUE

i Percent Toggle: FALSE

# A tibble: 4 x 4
  method paper_id    cs n_experts
  <chr>  <chr>    <dbl>     <int>
1 ArMean 108       74           5
2 ArMean 138       68.6         5
3 ArMean 186       57.6         5
4 ArMean 24        22.2         5
```

5.2. Comparing and Evaluating Aggregation Methods

In real study contexts, such as that of the repliCATS project in the DARPA SCORE program, it is of interest to compute Confidence Scores using multiple aggregation methods so that their performance might be evaluated and compared. Since different methods offer different mathematical properties, and therefore might be more or less appropriate depending on the purpose of the aggregation and forecasting, a researcher or analyst might want to check how the different assumptions embedded in different aggregation methods might influence the final Confidence Scores for a forecast – i.e. how robust are the results to different methods and therefore to different assumptions?

From a computational perspective, multiple aggregation methods must first be applied to the forecast prior to comparison and evaluation. This can be implemented very succinctly using **purrr**'s `map_dfr()` function (Henry and Wickham 2020) , which takes a list of aggregation methods, repeatedly applies each method to the dataframe `focal_claims`, and row-binds the resultant list of dataframes into a single dataframe, for example:

```
R> list(
+  AverageWAgg,
+  IntervalWAgg,
+  IntervalWAgg,
+  ShiftingWAgg,
+  BayesianWAgg
+) %>%
+  purrr::map2_dfr(.y = list("ArMean",
+                            "IndIntWAgg",
+                            "IntWAgg",
+                            "ShiftWAgg",
+                            "BayTriVar"),
+                  .f = ~ .x(focal_claims,
+                            type = .y,
+                            percent_toggle = TRUE)
+  )

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 20
   Unobserved stochastic nodes: 4
   Total graph size: 230

Initializing model

# A tibble: 20 x 4
   method     paper_id    cs n_experts
   <chr>      <chr>    <dbl>     <int>
 1 ArMean     108      0.74          5
 2 ArMean     138      0.686         5
 3 ArMean     186      0.576         5
 4 ArMean     24       0.222         5
 5 IndIntWAgg 108      0.740         5
 6 IndIntWAgg 138      0.685         5
 7 IndIntWAgg 186      0.561         5
 8 IndIntWAgg 24       0.19          5
 9 IntWAgg    108      0.748         5
10 IntWAgg    138      0.694         5
11 IntWAgg    186      0.581         5
12 IntWAgg    24       0.181         5
13 ShiftWAgg  108      0.715         5
14 ShiftWAgg  138      0.706         5
15 ShiftWAgg  186      0.438         5
16 ShiftWAgg  24       0.209         5
17 BayTriVar  108      0.699         5
18 BayTriVar  138      0.659         5
19 BayTriVar  186      0.528         5
20 BayTriVar  24       0.175         5
```

Given that aggregation methods `IntWAgg` and `IndIntWAgg` are both applied using the aggregation wrapper function `IntervalWAgg()`, but by supplying their method names as a character string to the type argument, we must supply a second list of character strings (the same length as our list of wrapper functions) to the mapping function. We therefore use `map2_dfr()` instead of `map_dfr()` because there are now multiple inputs that must be iterated along in parallel (the list of functions and the corresponding aggregation type) (Wickham and Grolemund 2017a).

Note that if we wish to batch aggregate claims using a combination of aggregation methods that do and do not require supplementary data, we must aggregate them separately, since the methods that require supplementary data have an additional argument for the supplementary data that must be parsed to the wrapper function call. We can chain the aggregation of the methods that do not require supplementary data, and the methods that do require supplementary data together very neatly using **dplyr**'s `bind_rows` function (Wickham, François, Henry, and Müller 2021) and the `magrittr()` pipe `%\>%` (Bache and Wickham 2020). Below we implement this approach while applying the aggregation methods `ArMean`, `IntWAgg`, `IndIntWAgg`, `ShiftingWAgg` and `BayTriVar` to the repliCATS pilot program dataset `data_ratings`:

```
R> confidenceSCOREs <-
+  list(
+    AverageWAgg,
+    IntervalWAgg,
+    IntervalWAgg,
+    ShiftingWAgg,
+    BayesianWAgg
+  ) %>%
+  purrr::map2_dfr(
+    .y = list("ArMean",
+              "IndIntWAgg",
+              "IntWAgg",
+              "ShiftWAgg",
+              "BayTriVar"),
+    .f = ~ .x(aggreCAT::data_ratings, type = .y, percent_toggle = TRUE)
+  ) %>%
+  dplyr::bind_rows(
+    ReasoningWAgg(aggreCAT::data_ratings,
+                  reasons = aggreCAT::data_supp_reasons,
+                  percent_toggle = TRUE)
+  )

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 625
   Unobserved stochastic nodes: 25
   Total graph size: 5904

Initializing model

R> confidenceSCOREs

# A tibble: 150 x 4
   method paper_id    cs n_experts
   <chr>  <chr>    <dbl>     <int>
 1 ArMean 100      0.706        25
 2 ArMean 102      0.308        25
 3 ArMean 103      0.625        25
 4 ArMean 104      0.471        25
 5 ArMean 106      0.365        25
 6 ArMean 108      0.718        25
 7 ArMean 109      0.725        25
 8 ArMean 116      0.626        25
 9 ArMean 118      0.548        25
10 ArMean 133      0.599        25
# ... with 140 more rows
```

After generating Confidence Scores using various aggregation methods, we then evaluate the forecasts. We evaluated the repliCATS pilot study forecasts against the outcomes of previous, high-powered replication studies (Hanea *et al.* 2021), which are contained in the data_outcomes dataset published with **aggreCAT**. In this dataset, each claim `paper_id` is assigned an `outcome` of `0` if the claim did not replicate and `1` if the claim was successfully replicated:

```
R> aggreCAT::data_outcomes %>%
+  head
```

```
# A tibble: 6 x 2
  paper_id outcome
  <chr>      <dbl>
1 100            1
2 102            0
3 103            0
4 104            1
5 106            0
6 108            1
```

The function `confidence_score_evaluation()` evaluates a set of aggregated forecasts or Confidence Scores against a set of known or observed outcomes, returning the Area Under the ROC Curve (AUC), the Brier score, and classification accuracy of each method (results displayed in Table 3):

Method	AUC	Brier Score	Classification Accuracy
ArMean	0.94	0.15	84%
BayTriVar	0.87	0.14	80%
IndIntWAgg	0.93	0.14	84%
IntWAgg	0.93	0.14	84%
ReasonWAgg	0.90	0.15	84%
ShiftWAgg	0.96	0.15	88%

Table 3: AUC and Classification Accuracy for the aggregation methods 'ShiftWAgg', 'ArMean', 'IntWAgg', 'IndIntWAgg', 'ReasonWAgg' and 'BayTriVar' evaluated for repliCATS pilot study claims and known outcomes.

5.3. Visualising Judgements, Confidence Scores and Forecast Performance

We include two functions for visualising comparison and eavluation of Confidence Scores across multiple aggregation methods for a suite of forecasts from multiple participants, `confidence_scores_ridgeplot()` and `confidencescore_heatmap()`. `confidence_scores_ridgeplot()` generates ridgeline plots using **ggridges** Wilke (2021) , and displays the distribution of predicted outcomes across a suite of forecasts for each aggregation method, grouped into separate 'mountain ranges' according to the mathematical properties of the aggregation method Figure 5.

While `confidencescore_heatmap()` is useful for comparison of aggregation methods, `confidencescore_heatmap()` is useful for visual comparative *evaluation* of aggregation methods. `confidencescore_heatmap()` generates heatmaps of forecasted Confidence Scores for each aggregation method included in the dataset provided to the argument confidence_scores organised with unique aggregation methods on the y-axis, and separate forecasts or `paper_ids` along the y-axis Figure 6. The heatmap is blocked vertically according to the mathematical characteristics of each aggregation method, and horizontally into two groups, according to the binary outcomes in data_outcomes.

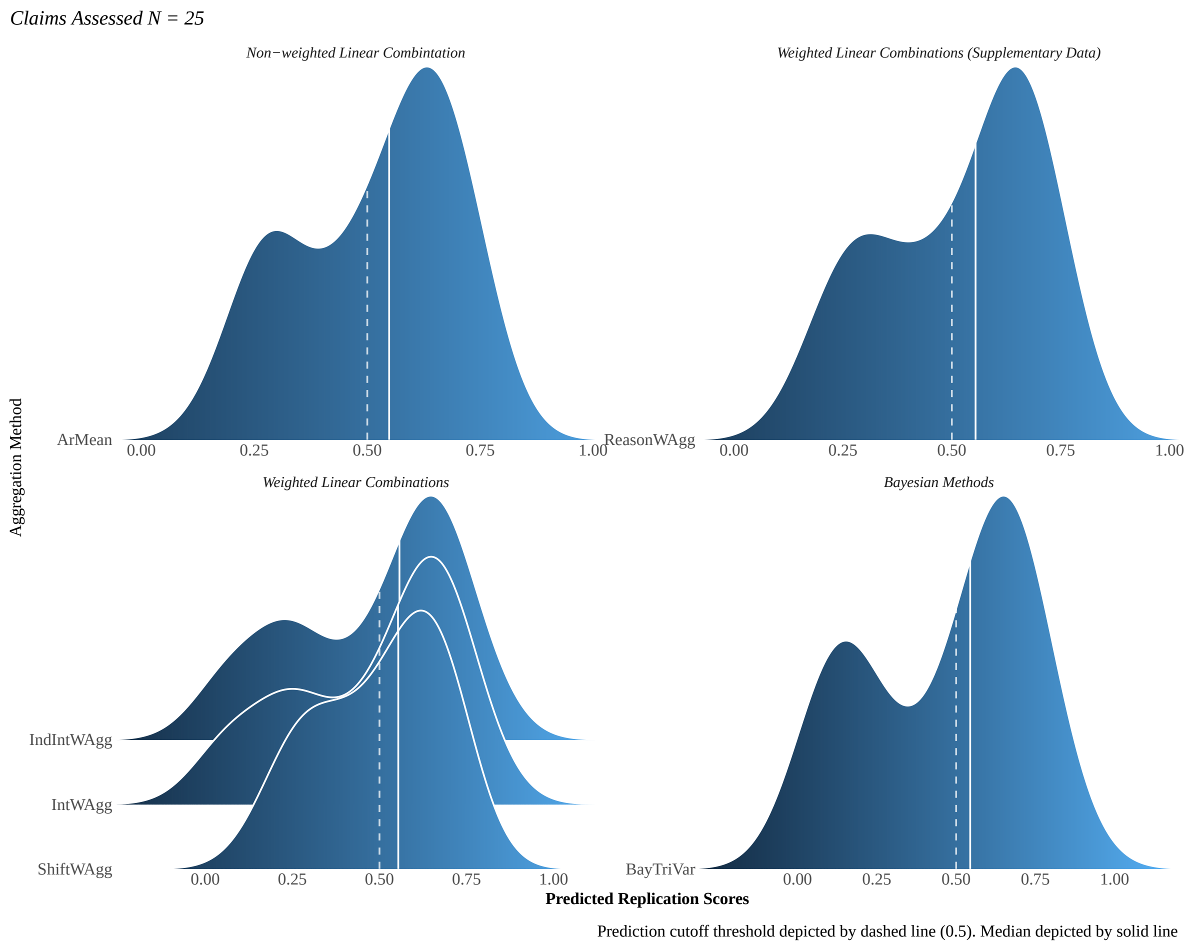


Figure 5: Ridge plots blah blah blah.

Horizontal grouping facilitates quick and simple evaluation of the aggregation methods. Perfectly accurate aggregation methods show dark blue squares in the left heatmap blocks, where the outcomes were `1` or `TRUE`, and dark red squares on the right heatmap blocks, where the actual outcomes were `0` or `FALSE`. Deviation from this expectation indicates which aggregation methods for which claim/forecast, for which outcome type were inaccurate, and to what degree.

For example, in figure Figure 6, for the example dataset confidenceSCOREs the successful replication of most claims was accurately forecasted by most methods, except for several claims. Some methods performed better than others for some claims (e.g. `BayTriVar` and `IndIntWAgg` for the first claim on the left (TODO insert), and for the claim on the right). In contrast, for most claims that did not replicate, forecasts were inaccurate, with `IndIntWAgg`, `IntWAgg` and `BayTriVar` performing particularly badly for the claims X and Y.

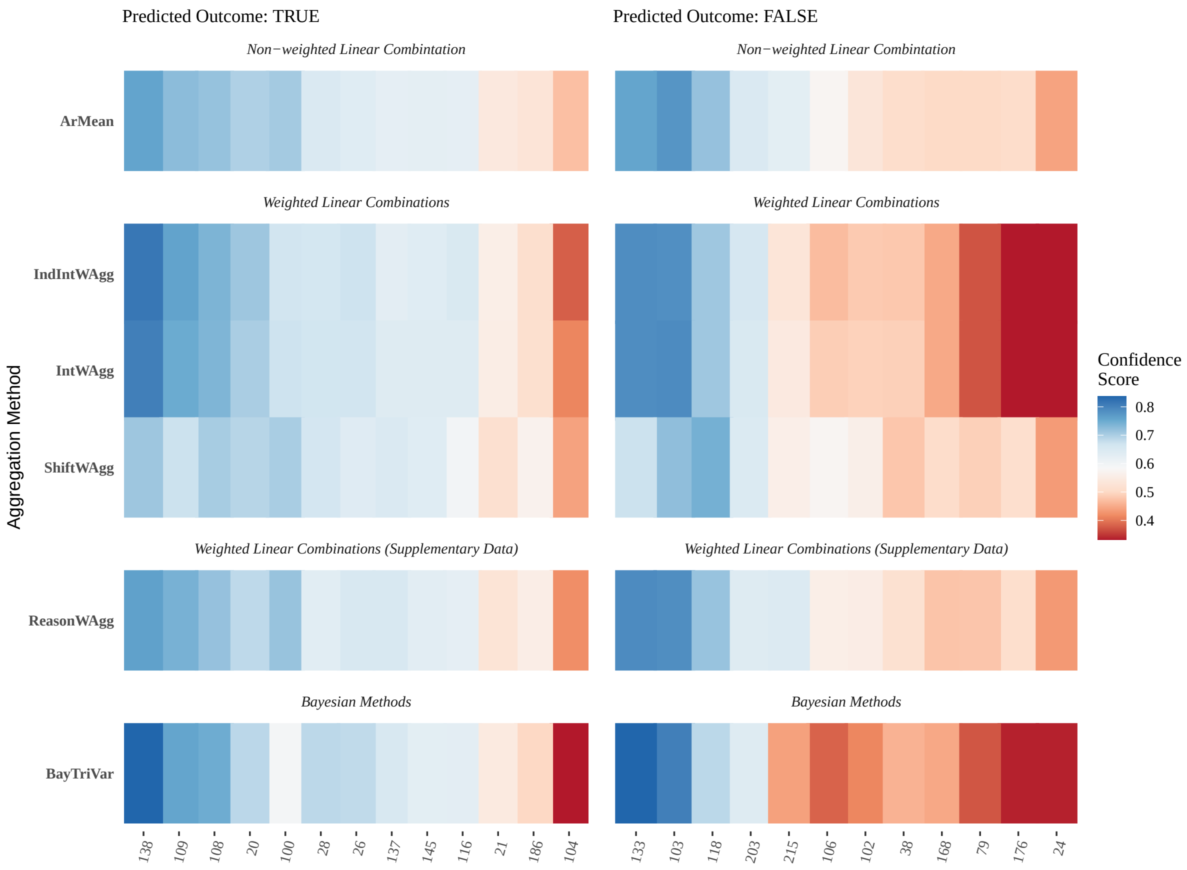


Figure 6: Blocked heatmap visualisation of confidence scores is useful for visually comparing aggregation methods and evaluating them against a set of known outcomes. In this example, Confidence Scores generated by 6 aggregation methods for the repliCATS pilot study are visualised for 25 claims. Claims where known outcomes succesfully replicated (outcome == `TRUE`) are presented in heatmap blocks on the left, and claims that failed to replicate are presented in heatmap blocks on the right. Confidence Scores generated by different aggregation methods are positioned along the y-axis, with vertical groupings according to the methods' mathematical properties. Colour and intensity of cells indicates the direction and degree of deviation respectively of the Confidence Scores from the known outcomes.

Finally, creating bespoke user-defined plots is relatively easy – because **aggreCAT** functions return tidy dataframes, we can easily manipulate the raw judgements, aggregated Confidence Scores and outcome data to plot them with **ggplot2** (Wickham 2016) or other visualisation package. Below we plot the aggregated Confidence Scores along with the three-point judgements (subset using `preprocess_judgements()` on focal_claims, transforming judgements in percentages to probabilities by setting percent_toggle to `TRUE`), Figure 7 :

#lst- identifier along with a lst-cap

```
plot_cs <-
  confidenceSCOREs %>%
  dplyr::left_join(aggreCAT::data_outcomes) %>%
  dplyr::mutate(data_type = "Confidence Scores") %>%
  dplyr::rename(x_vals = cs,
         y_vals = method) %>%
  dplyr::select(y_vals, paper_id, data_type, outcome, x_vals)
```

```
Joining, by = "paper_id"
```

```
plot_judgements <-
  aggreCAT::preprocess_judgements(focal_claims,
                                  percent_toggle = TRUE) %>%
  tidyr::pivot_wider(names_from = element,
                     values_from = value) %>%
  dplyr::left_join(aggreCAT::data_outcomes) %>%
  dplyr::rename(x_vals = three_point_best,
         y_vals = user_name) %>%
  dplyr::select(paper_id,
         y_vals,
         x_vals,
         tidyr::contains("three_point"),
         outcome) %>%
  dplyr::mutate(data_type = "Elicited Probabilities")
```

```
-- Pre-Processing Options --

i Round Filter: TRUE
i Three Point Filter: TRUE
i Percent Toggle: TRUE
Joining, by = "paper_id"
```

```
p <- plot_judgements %>%
  dplyr::bind_rows(., {dplyr::semi_join(plot_cs, plot_judgements,
                             by = "paper_id")}) %>%
  ggplot(aes(x = x_vals, y = y_vals)) +
  geom_pointrange(aes(xmin = three_point_lower,
                      xmax = three_point_upper)) +
  facet_grid(data_type ~ paper_id, scales = "free_y") +
  theme_classic() +
  theme(legend.position = "none") +
  geom_vline(aes(xintercept = 0.5, colour = as.logical(outcome))) +
  xlab("Probability of Replication") +
  ylab(element_blank()) +
  scale_colour_brewer(palette = "Set1")

p
```

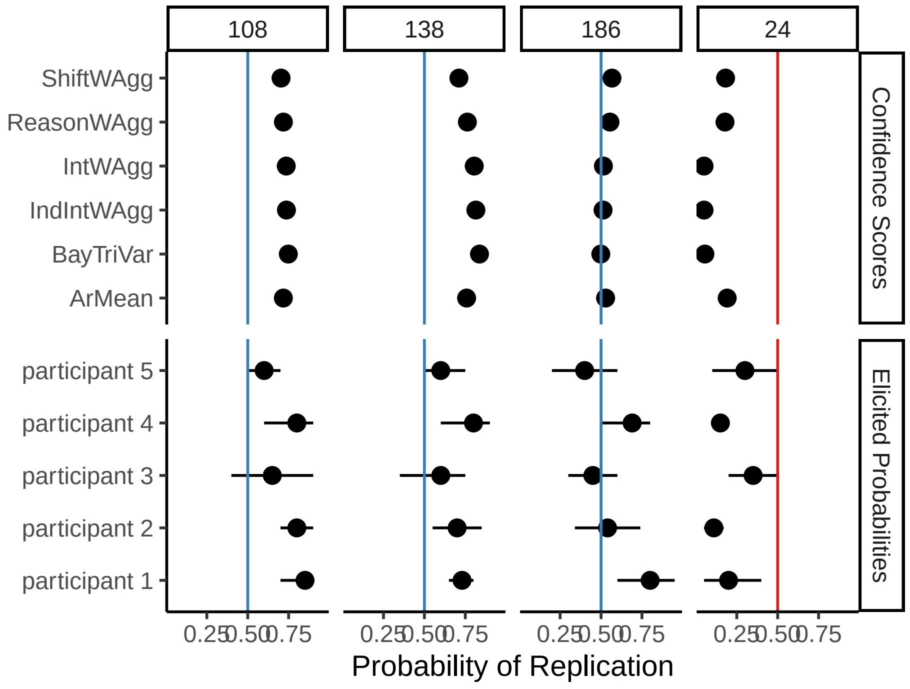


Figure 7: Confidence Scores for the aggregation methods ‘ArMean’, ‘BayTriVar’, ‘IntWAgg’, ‘IndIntWAgg’, ‘ReasonWAgg’ and ‘ShiftWAgg’ for four claims. Participants’ three-point best estimates are displayed as black points, and their upper and lowr bounds displayed as black error bars. Confidence Scores are displayed as points within the upper row of plots. Lines are displayed vertically at the 0.5 probability mark, and their colour denotes the observed outcome under previous large-scale replication projects.

5.4. Extending aggreCAT to other datasets

The aggregation methods contained in the **aggreCAT** package can easily be applied to other forecasting problems. The only requirements are that the data inputs adhere to the required format (see Box 1), and that the expert judgements are elicited using the appropriate method, as required by each aggregation method (see Table 1).

Judgement data provided to the `expert_judgements`, `data_justifications` or any supplementary data inputs argument must contain the requisite column names, and be of the correct data type, as described in each method's documentation (see `?data_ratings`, for example). At minimum the user must supply to `expert_judgements`: the `round` under which each judgement is elicited, a unique ID for each different forecasting problem `paper_id`, a unique `user_name` for each individual, and the `element` of the three point elicitation that the recorded response or `value` in that row corresponds to. The data is stored in long or tidy format such that each row or observation in the dataframe references only a single `element` of a participants' set of three point elicitation values. When applying aggregation methods requiring supplementary data to the elicitation data, the analyst should also adhere to the requirements stipulated for the relevant supplementary dataset described in the documentation.

Although several aggregation functions *require* judgements judgements are elicited using the IDEA protocol, most aggregation methods require only a single round of elicitation that generates a set of three points; a best estimate, and upper and lower bounds about those estimates. Hence, the aggregation functions contained in the **aggreCAT** package are unsuitable for use with judgements derived using Delphi or other similar elicitation methods that aggregate behaviourally (e.g. using consensus) and therefore result in a single forecast value. Where the analyst elicits judgements for only a single round, the analyst should record the round in the judgements data as the character string `round_2`, which is the default source of estimates for aggregation methods where only a single round of data is required, but where the IDEA protocol has been used to elicit judgements.

Should the analyst wish to create their own aggregation functions, pre- and post-processing functions may be leveraged inside the functions (`preprocess_judgements()` and `postprocess_judgements()`, respectively), as we have illustrated in data preparation for Figure 7, **?@lst-confidencescores**. These processing functions modularise key components of the aggregation's computational implementation - namely the data wrangling that occurs before and after the actual mathematical aggregation.

Preparing your own Elicitation Data

We demonstrate how to prepare data for applying the **aggreCAT** aggregation methods with data collected using the IDEA protocol for an environmental conservation problem (Arlidge 2020) . Participants were asked "How many green turtles in winter per month would be saved using a total gillnet ban, with gear switching to lobster potting or hand line fishing required?". We take the required data for the `expert_judgements` argument from Table S51 of Arlidge et al. (2020), make the data long instead of wide, and then add the required additional columns `paper_id` and `question`:

```
R> green_turtles <-
+  dplyr::tribble(~user_name, ~round, ~three_point_lower,
+          ~three_point_upper, ~three_point_best,
+          "L01", 1,    10.00,  16.43,  10.00,
+          "L01", 2,    10.00,  16.43,  10.00,
+          "L02", 1,    500.00, 522.50, 500.00,
+          "L02", 2,    293.75, 406.25, 350.00,
+           "L03", 1,    400.00, 512.50, 400.00,
+           "L03", 2,    300.00, 356.25, 300.00,
+           "L04", 1,    32.29,  65.10,  41.67,
+           "L04", 2,    32.29,  65.10,  41.67,
+           "L05", 1,    6.67,   7.74,   6.67,
+           "L05", 2,    6.67,   7.74,   6.67) %>%
+  dplyr::group_by(user_name) %>% # pivot longer
+  tidyr::pivot_longer(cols = tidyr::contains("three_point"),
+               names_to = "element", "value") %>%
+  dplyr::mutate(paper_id = 1,
+         round = ifelse(round ==1, "round_1", "round_2"),
+         question = "direct_replication")
```

We can then apply multiple aggregation methods, using the same approach implemented for aggregation of the focal_claims dataset (**?@lst-multi-method-workflow-non-supp**), with aggregated Confidence Scores shown in Table 4. Note that because the judgements are absolute values rather than probabilities, we set the percent_toggle argument for each aggregation wrapper function to FALSE:

```
R> turtle_CS <-
+  list(
+  AverageWAgg,
+  IntervalWAgg,
+  IntervalWAgg,
+  ShiftingWAgg
+) %>%
+  purrr::map2_dfr(.y = list("ArMean",
+                           "IndIntWAgg",
+                           "IntWAgg",
+                           "ShiftWAgg"),
+                 .f = ~ .x(green_turtles, type = .y,percent_toggle = FALSE)
+  )
```

Method	Question ID	Confidence Score	N (experts)
ArMean	1	141.67	5
IndIntWAgg	1	141.67	5
IntWAgg	1	15.26	5
ShiftWAgg	1	328.85	5

Table 4: Example aggregation of non-percentage / non-probabilistic estimates with several aggregation methods using Green Turtle dataset [@Arlidge2020].

5.5. TL;DR – Building reproducible workflows and dealing with regularly updated data

We have included several different types of functionality for when data collection is ongoing but where Confidence Scores need to be regularly aggregated for reporting; timestamp toggling, placeholder mode, and imputing

The date of the first and last assessment for any given claim may be computed by toggling on the logical argument, `timestamp`, within each aggregation function and by providing a timestamp for each three-point estimate value within `data_ratings`. The repliCATS elicitation platform automatically recorded the timestamp when a participant enters their assessment for a claim.

For most users of the the **aggreCAT** package , however, we anticipate that forecasts will be elicited during a single workshop before being aggregated on one instance. Under this use-case, providing information about the date and time of assessments is probably irrelevant. Consequently all post-processing and aggregation functions default to no timestamp functionality, and timestamps are not required to be included in the user's data parsed to the argument `expert_judgements` within the aggregation functions.

Similarly, when building a reproducible pipeline for working with regularly updated data (e.g. Yenni 2019), it can be useful to put aggregation methods into 'placeholder' mode, whereby a placeholder value is returned by the aggregation function instead of computing a Confidence Score using the aggregation method. This can be useful when developing unit-test code, or when modifying and testing a new workflow. For the repliCATS project, the placeholder was set and has been hard-coded in `method_placeholder()` to 0.65. Should the user wish to set an alternative value, they can create a modified version of `method_placeholder()` for themselves and store this within the global environment. This function will then be called by the aggregation method when the `placeholder` argument is set to `TRUE`.

Some aggregation methods default to the arithmetic mean of the log-odds transformed best estimate, i.e. `LoArMean()`, when the data requirements for that aggregation have not been met. For example, `reasonWAgg()` defaults to `LoArMean()` when no participants assessing a claim provided reasoning data. Instead of allowing this behaviour to occur silently, the user may wish to flag this behaviour explicitly by setting the argument `flag_loarmean` to `TRUE`, generating a new column in the aggregation output dataframe named `method_applied`. This column is a character vector consisting of either the name of the method called by the user where the conditions were satisfied, or `"LoArMean"` when the aggregator's conditions remain unsatisfied.

6. Discussion and Future Directions

The **aggreCAT** package provides a diverse suite of methods for mathematically aggregating judgements elicited from groups of experts using structured elicitation procedures, such as the IDEA protocol. The **aggreCAT** package was developed by the repliCATS project as a part of the DARPA SCORE program to implement the 28 aggregation methods described in Hanea et al. (2021).

There are very few open-source tools available to the researcher wishing to mathematically aggregate judgements. The **aggreCAT** package is therefore unique in both the diversity of aggregation methods it contains, as well as in its computational approach to implementing the aggregation methods. There is no other R or other software package with so many aggregation methods, and methods that use proxies of forecasting accuracy using weights.

The **aggreCAT** package is production-ready for application to data elicited during either a single workshop, or for production scenarios where continuous analysis is used and data collection is ongoing. Unlike other aggregation packages, the **aggreCAT** package is designed to work within the *tidyverse*. The package is premised on the principles of *tidy* data analysis whereby the user supplies dataframes of elicited judgements, and the aggregation methods return dataframes of aggregated forecasts. The benefits of this approach are three-fold. Firstly, the work of data-wrangling and application of the aggregation methods is handled internally by the aggregation methods, so that the researcher can focus on analysis and interpretation of the aggregation outputs. This is critical in data-deficient contexts where rapid assessments are needed, which is a common use-case for the use of expert derived forecasts. Secondly, the **aggreCAT** package is easily paired with other tidyverse tools, such as **purrr**, **dplyr**, and **ggplot2**, as exemplified through the repliCATS workflow described in section X.

Thirdly, application of the **aggreCAT** package aggregation methods and performance evaluation tools is scalable, which is evidenced by the application of the **aggreCAT** package to forecast the replicability of over 3000 research claims by the repliCATS project during phase 1 of the SCORE program. The scalability, timestamp and placeholder functionality allow the **aggreCAT** package to be built into production-ready pipelines for more complicated analyses where there are multiple forecasts being elicited and aggregated, where there are numerous participants, and where multiple aggregation methods are applied.

Finally, through the provision of built-in performance metrics, the analyst is able to 'ground-truth' and evaluate the forecasts against known-outcomes, or alternative forecasting methods (e.g. Arlidge 2020).

The **aggreCAT** package is easily extensible and production-ready. Each aggregation function follows a consistent modular blueprint, wherein data-wrangling of the inputs and outputs of aggregation is largely handled by pre- and post-processing functions (`preprocess_judgements()` and `postprocess_judgements()`, respectively). This design expedites debugging by making it easier to pinpoint the exact source of errors, while also permitting the user to easily create their own custom aggregation methods.

Although the package currently requires data inputs to conform to nomenclature specific to the repliCATS project, future releases of the **aggreCAT** package will relax the data-input requirements so they are more domain-agnostic. We believe this to be a minimal barrier for adoption and application of the **aggreCAT** package. Ecologists should be no stranger to these naming conventions for data requirements, with packages like **vegan** also imposing strict nomenclature (Oksanen, Blanchet, Friendly, Kindt, Legendre, McGlinn, Minchin, O'Hara, Simpson, Solymos, Stevens, Szoecs, and Wagner 2020). We have illustrated how to extend and apply the package to data from domains beyond forecasting the replicability of research claims through our minimal example using forecasts generated using the IDEA protocol for a fisheries and conservation problem.

The package will be actively maintained into the future, and we expect additional aggregation methods to be added to the package during phase 2 of the DARPA SCORE program. Bug reports and feature-requests can easily be lodged on the **aggreCAT** GitHub repository using reproducible examples created with **reprex** (Bryan, Hester, Robinson, and Wickham 2021) on the repliCATS pilot study datasets shipped with the **aggreCAT** package.

We have described the computational implementation of the aggregation methods and supporting tools within the **aggreCAT** package, providing usage examples and workflows for both simple and more complex research contexts. Consequently, this paper should fully equip the analyst for applying the aggregation functions contained within the **aggreCAT** package to their own data. Where the analyst is uncertain as to *which* aggregation method is best for their particular research goals, the reader should consult Hanea et al. (2021) for a discussion on the mathematical principles and hypotheses underlying the design of the aggregation methods, as well as a comparative performance evaluation of each of the methods. In conclusion, the **aggreCAT** package will aid researchers and decision analysts in rapidly and easily analysing the results of IDEA protocol and other structured elicitation procedures where mathematical aggregation of human forecasts is required.

> TODO: Note that around the equation above there should be no spaces (avoided in the LaTeX code by `%` lines) so that "normal" spacing is used and not a new paragraph started.

R provides a very flexible implementation of the general GLM framework in the function `glm()` Chambers and Hastie (1992) in the **stats** package. Its most important arguments are

```
glm(formula, data, subset, na.action, weights, offset,
  family = gaussian, start = NULL, control = glm.control(…),
  model = TRUE, y = TRUE, x = FALSE, …)
```

where `formula` plus `data` is the now standard way of specifying regression relationships in R/S introduced in Chambers and Hastie (1992). The remaining arguments in the first line (`subset`, `na.action`, `weights`, and `offset`) are also standard for setting up formula-based regression models in R/S. The arguments in the second line control aspects specific to GLMs while the arguments in the last line specify which components are returned in the fitted model object (of class '`glm`' which inherits from '`lm`'). For further arguments to `glm()` (including alternative specifications of starting values) see `?glm`. For estimating a Poisson model `family = poisson` has to be specified.

More technical details

> Appendices can be included after the bibliography (with a page break). Each section within the appendix should have a proper section title (rather than just *Appendix*).
> For more technical style details, please check out JSS's style FAQ at [https://www.jstatsoft.org/pages/view/style#frequently-asked-questions] which includes the following topics:
>
> - Title vs. sentence case.

- Graphics formatting.
- Naming conventions.
- Turning JSS manuscripts into R package vignettes.
- Trouble shooting.
- Many other potentially helpful details…

Using BibTeX

References need to be provided in a BibTeX file (`.bib`). All references should be made with `@cite` syntax. This commands yield different formats of author-year citations and allow to include additional details (e.g.,pages, chapters, …) in brackets. In case you are not familiar with these commands see the JSS style FAQ for details.
Cleaning up BibTeX files is a somewhat tedious task – especially when acquiring the entries automatically from mixed online sources. However, it is important that informations are complete and presented in a consistent style to avoid confusions. JSS requires the following format.

- item JSS-specific markup (`\proglang`, `\pkg`, `\code`) should be used in the references.
- item Titles should be in title case.
- item Journal titles should not be abbreviated and in title case.
- item DOIs should be included where available.
- item Software should be properly cited as well. For R packages `citation("pkgname")` typically provides a good starting point.

7. Summary and discussion

Computational details

If necessary or useful, information about certain computational details such as version numbers, operating systems, or compilers could be included in an unnumbered section. Also, auxiliary packages (say, for visualizations, maps, tables, …) that are not cited in the main text can be credited here.

The results in this paper were obtained using R~3.4.1 with the **MASS**~7.3.47 package. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at [https://CRAN.R-project.org/].

```
sessionInfo()
```

```
R version 4.2.0 (2022-04-22)
```

```
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Big Sur/Monterey 10.16

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib

locale:
[1] en_AU.UTF-8/en_AU.UTF-8/en_AU.UTF-8/C/en_AU.UTF-8/en_AU.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods
[7] base

other attached packages:
 [1] ggpubr_0.4.0        ggforce_0.3.3       ggridges_0.5.3
 [4] aggreCAT_0.0.0.9001 forcats_0.5.1       stringr_1.4.1
 [7] dplyr_1.0.10        purrr_0.3.4         readr_2.1.2
[10] tidyr_1.2.0         tibble_3.1.8        ggplot2_3.3.6
[13] tidyverse_1.3.1

loaded via a namespace (and not attached):
 [1] fs_1.5.2           lubridate_1.8.0    RColorBrewer_1.1-3
 [4] insight_0.18.2     httr_1.4.4         tools_4.2.0
 [7] backports_1.4.1    utf8_1.2.2         R6_2.5.1
[10] R2WinBUGS_2.1-21   DBI_1.1.3          colorspace_2.0-3
[13] withr_2.5.0        gridExtra_2.3      tidyselect_1.1.2
[16] Exact_3.1          compiler_4.2.0     cli_3.3.0
[19] rvest_1.0.2        gt_0.6.0.9000      expm_0.999-6
[22] xml2_1.3.3         labeling_0.4.2     scales_1.2.1
[25] mvtnorm_1.1-3      proxy_0.4-27       digest_0.6.29
[28] rmarkdown_2.14.3   pkgconfig_2.0.3    htmltools_0.5.3
[31] dbplyr_2.2.0       fastmap_1.1.0      rlang_1.0.5
[34] readxl_1.4.0       rstudioapi_0.13    generics_0.1.3
[37] farver_2.1.1       jsonlite_1.8.0     car_3.1-0
[40] magrittr_2.0.3     Matrix_1.4-1       Rcpp_1.0.9
[43] DescTools_0.99.45  munsell_0.5.0      fansi_1.0.3
[46] abind_1.4-5        lifecycle_1.0.1    stringi_1.7.8
[49] yaml_2.3.5         carData_3.0-5      MASS_7.3-56
[52] rootSolve_1.8.2.3  plyr_1.8.7         grid_4.2.0
[55] parallel_4.2.0     crayon_1.5.1       lmom_2.9
[58] lattice_0.20-45    cowplot_1.1.1      haven_2.5.0
[61] hms_1.1.1          knitr_1.39         pillar_1.8.1
[64] boot_1.3-28        gld_2.6.4          ggsignif_0.6.3
[67] reprex_2.0.1       rfUtilities_2.1-5  precrec_0.12.9
[70] R2jags_0.7-1       glue_1.6.2         evaluate_0.15
[73] data.table_1.14.2  modelr_0.1.8       tweenr_1.0.2
```

```
[76] png_0.1-7          vctrs_0.4.1        tzdb_0.3.0
[79] cellranger_1.1.0   polyclip_1.10-0    gtable_0.3.1
[82] assertthat_0.2.1   xfun_0.31          broom_0.8.0
[85] e1071_1.7-11       rstatix_0.7.0      coda_0.19-4
[88] class_7.3-20       rjags_4-13         ellipsis_0.3.2
```

Acknowledgments

(2015). *Estimating the reproducibility of psychological science*, volume 349(6251). American Association for the Advancement of Science.

(2021). *Eliciting group judgements about replicability: a technical implementation of the IDEA Protocol*. Hawaii International Conference on System Sciences. URL http://hdl.handle.net/10125/70666.

Alipourfard N, Arendt B, Benjamin DM, Benkler N, Bishop MM, Burstein M, Bush M, Caverlee J, Chen Y, Clark C, et al (2021). "Systematizing Confidence in Open Research and Evidence (SCORE)." `doi:10.31235/osf.io/46mnb`. URL osf.io/preprints/socarxiv/46mnb.

Arlidge WNSea (2020). "Evaluating elicited judgments of turtle captures for data-limited fisheries management." *Conservation Science and Practice*, **2**(5).

Bache SM, Wickham H (2020). *magrittr: A Forward-Pipe Operator for R*. R package version 2.0.1, URL https://CRAN.R-project.org/package=magrittr.

Bryan J, Hester J, Robinson D, Wickham H (2021). *reprex: Prepare Reproducible Example Code via the Clipboard*. R package version 2.0.0, URL https://CRAN.R-project.org/package=reprex.

Camerer Cea (2018). "Evaluating the replicability of social science experiments in Nature and Science between 2010 and 2015." *naturecom*.

Chambers JM, Hastie TJ (eds.) (1992). *Statistical Models in S*. Chapman & Hall, London.

Ebersole CRea (2016). "Many Labs 3: Evaluating participant pool quality across the academic semester via replication." *Journal of Experimental Social Psychology*, **67**, 68–82.

Fraser H, Bush M, Wintle B, Mody F, Smith ET, Hanea A, Gould E, Hemming V, Hamilton DG, Rumpff L, et al (2021). "Predicting reliability through structured expert elicitation with repliCATS (Collaborative Assessments for Trustworthy Science)." `doi:10.31222/osf.io/2pczv`. URL osf.io/preprints/metaarxiv/2pczv.

Goossens Lea (2008). "Fifteen years of expert judgement at TUDelft." *Safety Science*, **46**(2), 234–244.

Gordon Mea (2020). "Are replication rates the same across academic fields? Community forecasts from the DARPA SCORE programme." *R Soc open sci*, **7**(7), 200566.

Gould E, Willcox A, Fraser H, Singleton Thorn F, Wilkinson DP (2021). "Using model-based predictions to inform the mathematical aggregation of human-based predictions of replicability." `doi:10.31222/osf.io/f675q`. URL osf.io/preprints/metaarxiv/f675q.

Hanea A, Wilkinson DP, McBride M, Lyon A, van Ravenzwaaij D, Singleton Thorn F, Gray CT, Mandel DR, Willcox A, Gould E, et al (2021). "Mathematically aggregating experts' predictions of possible futures." *PLoS ONE*, **16**(9). `doi:https://doi.org/10.1371/journal.pone.0256919`. URL https://doi.org/10.1371/journal.pone.0256919.

Hemming V, Burgman M, Hanea A, McBride M, Wintle B (2017). "A practical guide to structured expert elicitation using the IDEA protocol." *Methods in Ecology and Evolution*, **9**(1), 169–180. `doi:10.1111/2041-210x.12857`. URL http://dx.doi.org/10.1111/2041-210x.12857.

Henry L, Wickham H (2020). *purrr: Functional Programming Tools*. R package version 0.3.4, URL https://CRAN.R-project.org/package=purrr.

Isager PMea (2020). "Deciding what to replicate: A formal definition of "replication value" and a decision model for replication study selection." *Journal of Informetrics*, **13**(2), 635–642.

Klein RA, Vianello M, Hasselman F, Adams B, Adams R, Alper S, Aveyard M, Axt JR, Babalola MT, Bahník Š, Batra R, Berkics M, Bernstein M, Berry DR, Bialobrzeska O, Binan ED, Bocian K, Brandt M, Busching R, Redei A, Cai H, Cambier F, Cantarero K, Carmichael CL, Céric F, Chandler JJ, Chang J, Chatard A, Chen E, Cheong W, Cicero D, Coen S, Coleman JA, Collisson B, Conway M, Corker K, Curran P, Cushman F, Dagona Z, Dalgar I, Rosa A, Davis W, Bruijn MD, Schutter LD, Devos T, Vries MD, Doğulu C, Dozo N, Dukes K, Dunham Y, Durrheim K, Ebersole C, Edlund J, Eller A, English AS, Finck C, Frankowska N, Freyre M, Friedman M, Galliani E, Gandi JC, Ghoshal T, Giessner S, Gill T, Gnambs T, Gómez Á, Gonzalez R, Graham J, Grahe JE, Grahek I, Green E, Hai K, Haigh M, Haines EL, Hall MP, Heffernan ME, Hicks J, Houdek P, Huntsinger JR, Huynh H, Ijzerman H, Inbar Y, Innes-Ker Å, Jimenez-Leal W, John MS, Joy-Gaba JA, Kamiloğlu R, Kappes H, Karabati S, Karick H, Keller VN, Kende A, Kervyn N, Knežević G, Kovacs C, Krueger LE, Kurapov G, Kurtz J, Lakens D, Lazarević LB, Levitan C, Lewis N, Lins S, Lipsey NP, Losee JE, Maassen E, Maitner AT, Malingumu W, Mallett RK, Marotta SA, Međedović J, Mena-Pacheco F, Milfont T, Morris WL, Murphy S, Myachykov A, Neave N, Neijenhuijs K, Nelson AJ, Neto F, Nichols AL, Ocampo A, O'Donnell S, Oikawa H, Oikawa M, Ong E, Orosz G, Osowiecka M, Packard G, Pérez-Sánchez R, Petrović B, Pilati R, Pinter B, Podesta L, Pogge G, Pollmann M, Rutchick AM, Saavedra P, Saeri AK, Salomon E, Schmidt K, Schönbrodt FD, Sekerdej M, Sirlopú D, Skorinko JLM, Smith MA, Smith-Castro V, Smolders K, Sobkow A, Sowden W, Spachtholz P, Srivastava M, Steiner TG, Stouten J, Street CNH, Sundfelt OK, Szeto S, Szumowska E, Tang ACW, Tanzer NK, Tear MJ, Theriault JE, Thomae M, Torres D, Traczyk J, Tybur JM, Ujhelyi A, Aert RCM, Assen MV, van der Hulst M, Lange PV, Veer AEV, Echeverría A, Vaughn L, Vázquez A, Vega LD, Verniers C, Verschoor M, Voermans I, Vranka M, Welch CA, Wichman A, Williams L, Wood M, Woodzicka JA, Wronska MK, Young L, Zelenski J, Zhi-jia Z, Nosek BA (2018). "Many Labs 2: Investigating Variation in Replicability Across Samples and Settings." *Advances in Methods and Practices in Psychological Science*, **1**, 443 – 490.

Klein RAea (2014). "Investigating Variation in Replicability." *Social Psychology*, **45**(3), 142–152.

Oksanen J, Blanchet FG, Friendly M, Kindt R, Legendre P, McGlinn D, Minchin PR, O'Hara RB, Simpson GL, Solymos P, Stevens MHH, Szoecs E, Wagner H (2020). *vegan: Community Ecology Package*. R package version 2.5-7, URL https://CRAN.R-project.org/package=vegan.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Su YS, Yajima M (2020). *R2jags: Using R to Run 'JAGS'*. R package version 0.6-1, URL https://CRAN.R-project.org/package=R2jags.

Sutherland WJea (2018). "Qualitative methods for ecologists and conservation scientists." *Methods in Ecology and Evolution*, **9**(1), 7–9.

Wickham H (2014). "Tidy data." *Journal of Statistical Software*, **59**(10).

Wickham H (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org.

Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to the tidyverse." *Journal of Open Source Software*, **4**(43), 1686. `doi:10.21105/joss.01686`.

Wickham H, François R, Henry L, Müller K (2021). *dplyr: A Grammar of Data Manipulation*. R package version 1.0.6, URL https://CRAN.R-project.org/package=dplyr.

Wickham H, Grolemund G (2017a). "Chapter 17: Iteration with Purr." In *R for Data Science*, pp. 313–344. O'Reilly, Sebastpool, Canada.

Wickham H, Grolemund G (2017b). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. 1 edition. O'Reilly Media. ISBN 1491910399. URL http://r4ds.had.co.nz/.

Wilke CO (2021). *ggridges: Ridgeline Plots in 'ggplot2'*. R package version 0.5.3, URL https://CRAN.R-project.org/package=ggridges.

Wintle B (2021). "Predicting and reasoning about replicability using structured groups." *TBD*.

Yenni Gea (2019). "Developing a modern data workflow for regularly updated data." *PLoS Biol*, **17**(1), e3000125.

Affiliation:

Elliot Gould[3]
E-mail: `elliot.gould (at) unimelb.edu.au`

Charles Gray

Aaron Willcox

Rose O'Dea

Rebecca Groenewegen

David P. Wilkinson

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
`doi:10.18637/jss.v000.i00` *Accepted:* yyyy-mm-dd

[3]School of Ecosystem and Forest Sciences, University of Melbourne