

Multi-Model Ensembles in Infectious Disease and Public Health: Methods, Interpretation, and Implementation in R

Li Shandross¹ | Emily Howerton² | Lucie Contamin³ | Harry Hochheiser³ | Anna Krystalli⁴ | Consortium of Infectious Disease Modeling Hubs | Nicholas G. Reich¹ | Evan L. Ray¹

¹Department of Biostatistics and Epidemiology, University of Massachusetts Amherst, Amherst, Massachusetts, USA | ²Department of Biology, Center for Infectious Disease Dynamics, The Pennsylvania State University, State College, Pennsylvania, USA | ³Department of Biomedical Informatics, University of Pittsburgh, Pittsburgh, Pennsylvania, USA | ⁴R-RSE SMPC, Syros, Greece

Correspondence: Emily Howerton (ehowerton@psu.edu)

Received: 29 April 2025 | **Revised:** 6 October 2025 | **Accepted:** 17 November 2025

For a complete list of the Consortium of Infectious Disease Modeling Hubs, see the Acknowledgments section.

Keywords: aggregation | forecast | multiple models | prediction

ABSTRACT

Combining predictions from multiple models into an ensemble is a widely used practice across many fields with demonstrated performance benefits. Popularized through domains such as weather forecasting and climate modeling, multi-model ensembles are becoming increasingly common in public health and biological applications. For example, multi-model outbreak forecasting provides more accurate and reliable information about the timing and burden of infectious disease outbreaks to public health officials and medical practitioners. Yet, understanding and interpreting multi-model ensemble results can be difficult, as there are a diversity of methods proposed in the literature with no clear consensus on which is best. Moreover, a lack of standard, easy-to-use software implementations impedes the generation of multi-model ensembles in practice. To address these challenges, we provide an introduction to the statistical foundations of applied probabilistic forecasting, including the role of multi-model ensembles. We introduce the `hubEnsembles` package, a flexible framework for ensembling various types of predictions using a range of methods. Finally, we present a tutorial and case-study of ensemble methods using the `hubEnsembles` package on a subset of real, publicly available data from the FluSight Forecast Hub.

1 | Introduction

Predictions of future outcomes are essential to planning and decision making, yet generating reliable predictions of the future is challenging. One method for overcoming this challenge is combining predictions across multiple, independent models. These combination methods (also called aggregation or ensembling) have been repeatedly shown to produce predictions that are more accurate [1, 2] and more consistent [3] than individual models.

Because of the clear performance benefits, multi-model ensembles are a widely used statistical tool across fields, including weather forecasting [4], climate modeling [5], and economics [6]. In the last decade, the number of multi-model ensemble predictions generated and used in real time for public health planning and response has grown rapidly.

In particular, predicting infectious disease outbreaks and anticipating the effects of potential interventions has demonstrated

Li Shandross and Emily Howerton contributed equally to this study.

This is an open access article under the terms of the [Creative Commons Attribution](#) License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2026 The Author(s). *Statistics in Medicine* published by John Wiley & Sons Ltd.

utility for public health officials and medical practitioners. Underlying these predictions are mathematical models that use historical disease incidence data, including cases, hospitalizations, and deaths, to make probabilistic predictions of incidence in the future [7–11]. Given the performance benefits of multi-model ensembles, it is becoming increasingly common to convene multiple modeling teams into a collaborative “hub” [12], where each team generates independent predictions that are aggregated to collectively produce an ensemble. For example, this approach has been used to make real-time, multi-model predictions for seasonal influenza [10, 13], dengue [14], West Nile virus [15], and more recently SARS-CoV-2 [16–18].

Generating multi-model ensembles or interpreting the resulting predictions depends on understanding the underlying statistical methodology. There are many proposed methods for generating ensembles, and these methods differ in at least one of two ways: (1) the function used to combine or “average” predictions, and (2) how predictions are weighted when performing the combination. A few methodological papers have discussed theory of multi-model ensembles and tested various ensembling methods in public health applications specifically [19–25], yet there is no consensus on which method should be favored. There are software packages that support various aspects of multi-model ensembling [26–29]. However, these packages only support a subset of methods and prediction types, illustrating the need for standard, easy-to-use implementations of common methods.

Here, we provide an introduction to the statistical foundations of multi-model ensembles in applied probabilistic forecasting (Section 2). In addition, to improve accessibility, reproducibility, and interoperability, we have developed a comprehensive R package `hubEnsembles` that implements these methods (Section 3). The `hubEnsembles` package provides a flexible framework for generating ensemble predictions from multiple models across a range of common methods and prediction types, including both discrete and continuous outcomes. It is situated within the broader “hubverse” collection of open-source software and data tools to facilitate the development and management of collaborative modeling exercises [30]. These two factors together, a simple implementation framework across methods and integration with hubverse data standards and tools, make `hubEnsembles` accessible and easy to use.

Finally, we present a basic demonstration of multi-model ensemble generation and interpretation (Section 4), and a more in-depth analysis using real influenza forecasts (Section 5). The case studies demonstrate the utility of `hubEnsembles` to support a range of prediction types, and together motivate a discussion and comparison of the various methods (Section 6). While the case studies focus on infectious disease applications, the software and tools presented are general and could be used for applications in other areas of biomedical and public health research, or other domains. By reviewing multi-model ensemble methodology and synthesizing these methods into an easy-to use implementation, this tutorial provides guidance on understanding, interpreting, and implementing multi-model ensembles.

2 | How to Generate a Multi-Model Ensemble

In this section, we provide an overview of the process to generate a multi-model ensemble, including an outline of key statistical

concepts in probabilistic forecasting, and an overview of the classes of methods that are typically used for generating a multi-model ensemble. See Box 1 for a glossary of key terms and definitions.

2.1 | Key Statistical Concepts in Forecasting

Generating an ensemble requires multiple predictions to be combined, and a combination method for calculating the ensemble from these predictions (Figure 1). These predictions will often be produced by different statistical or mathematical models, and the output from these models (referred to as “model output” from here on) will vary based on the setting. For example, some public health questions, such as short-term resource allocation, may depend on a forecast of public health outcomes weeks into the future, whereas longer-term decisions about vaccination schedules may require projections months into the future across multiple possible scenarios. Some intervention decisions, such as quarantine and isolation policy, may depend on estimates of key biological parameters such as the generation interval for an infectious pathogen. Throughout, we will use the general term “prediction” to encapsulate all such outcomes that could be modeled, encompassing short-term forecasts, scenario projections, and parameter estimates. Predictions can also capture varying degrees of uncertainty in the outcome. A *point prediction* gives a single estimate of an outcome while a *probabilistic prediction* provides an estimated probability distribution over a set of outcomes. In either case, the basic steps required to generate an ensemble are the same.

2.2 | Mathematical Definitions and Properties of Ensemble Methods

Here, we use N to denote the total number of individual predictions that the ensemble will combine. For example, if predictions are produced by different models, N is the total number of models that have provided predictions. Individual predictions will be indexed by the subscript i . Optionally, one can calculate an ensemble that uses a weight w_i for each prediction; we define the set of model-specific weights as $\mathbf{w} = \{w_i | i \in 1, \dots, N\}$. Informally, predictions with a larger weight have a greater influence on the ensemble prediction, though the details of this depend on the ensemble method (described further below).

Then, for a set of N point predictions, $\mathbf{p} = \{p_i | i \in 1, \dots, N\}$, each from a distinct model i , an ensemble of these predictions is

$$p_E = C(\mathbf{p}, \mathbf{w})$$

using any function C and any set of model-specific weights \mathbf{w} . For example, an arithmetic average of predictions yields $p_E = \sum_{i=1}^N p_i w_i$, where the weights are non-negative and sum to 1. If $w_i = 1/N$ for all i , all predictions will be equally weighted. More complex functions for aggregation are also possible, such as a (weighted) median or geometric mean.

For probabilistic predictions, there are two commonly used classes of methods to average or ensemble multiple predictions: quantile averaging (also called a Vincent average [31]) and probability averaging (also called a distributional mixture

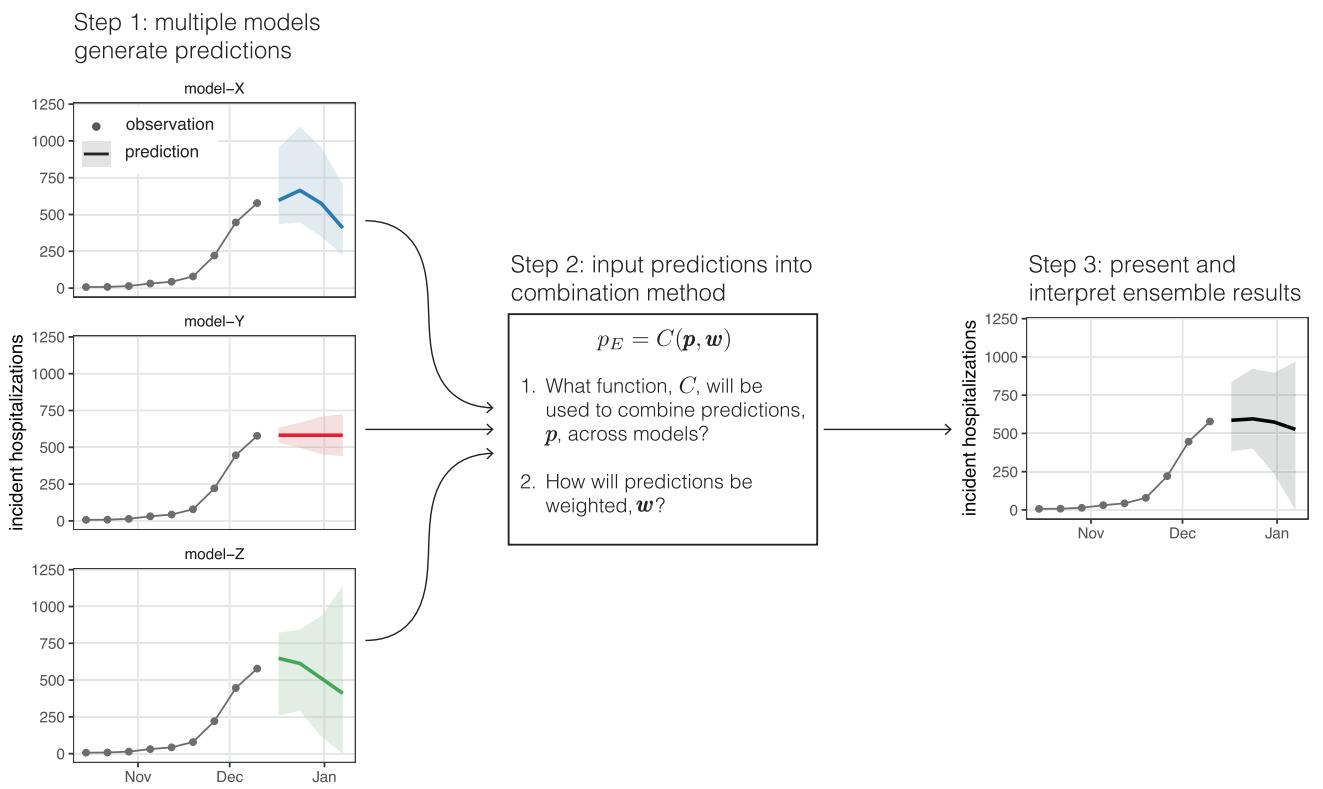


FIGURE 1 | Overview of process to generate a multi-model ensemble. Predictions, p_i are generated from N independent models (step 1). Then those predictions, $\mathbf{p} = \{p_i | i \in 1, \dots, N\}$, are combined with some function, C , and set of weights, $\mathbf{w} = \{w_i | i \in 1, \dots, N\}$. This figure illustrates example probabilistic forecasts for incident influenza hospitalizations, where the median (line) and 90% prediction interval are shown. In this case, the ensemble is constructed using the linear pool method ($F_{LOP}(x)$).

or linear opinion pool [32]) [33]. To define these two classes of methods, let $F(x)$ be a cumulative density function (CDF) defined over values x of the target variable for the prediction, and $F^{-1}(\theta)$ be the corresponding quantile function defined over quantile levels $\theta \in [0, 1]$. Throughout this article, we may refer to x as either a “value of the target variable” or a “quantile” depending on the context, and similarly we may refer to θ as either a “quantile level” or a “(cumulative) probability”. Additionally, we will use $f(x)$ to denote a probability mass function (PMF) for a prediction of a discrete variable or a discretization (such as binned values) of a continuous variable.

The quantile average combines a set of quantile functions, $Q = \{F_i^{-1}(\theta) | i \in 1, \dots, N\}$, with a given set of weights, \mathbf{w} , as

$$F_Q^{-1}(\theta) = C_Q(Q, \mathbf{w}) = \sum_{i=1}^N w_i F_i^{-1}(\theta).$$

This computes the average value of predictions across different models for each fixed quantile level θ . For a normal distribution or any distribution with a location and scale parameter, the resulting quantile average will be the same type of distribution, with location and scale parameters that are the average of the location and scale parameters from the individual distributions (Figure 2, panel B). In other words, this method interprets the predictive probability distributions that are being combined as uncertain estimates of a single true distribution. It is also possible to use

other combination functions, such as a weighted median, to combine quantile predictions.

The probability average or linear opinion pool (LOP, or simply linear pool) is calculated by averaging probabilities across predictions for a fixed value of the target variable, x . In other words, for a set of CDFs $\mathcal{F} = \{F_i(x) | i \in 1, \dots, N\}$ and weights \mathbf{w} , the linear pool is calculated as

$$F_{LOP}(x) = C_{LOP}(\mathcal{F}, \mathbf{w}) = \sum_{i=1}^N w_i F_i(x).$$

For a set of PMF values, $\{f_i(x) | i \in 1, \dots, N\}$, the linear pool can be equivalently calculated as $f_{LOP}(x) = \sum_{i=1}^N w_i f_i(x)$. Statistically this amounts to a mixture of the probability distributions, and the resulting probability distribution can be interpreted as one where the constituent probability distributions represent alternative predictions of the future, each of which has a probability w_i of being the true one. If individual samples (trajectories) from the predictive distributions are collected, the LOP is equivalent to collecting samples across all models and pooling them into a single distribution. For a visual depiction of these equations, see Figure 2 below.

The different averaging methods for probabilistic predictions yield different properties of the resulting ensemble distribution. For example, the variance of the linear pool is $\sigma_{LOP}^2 =$

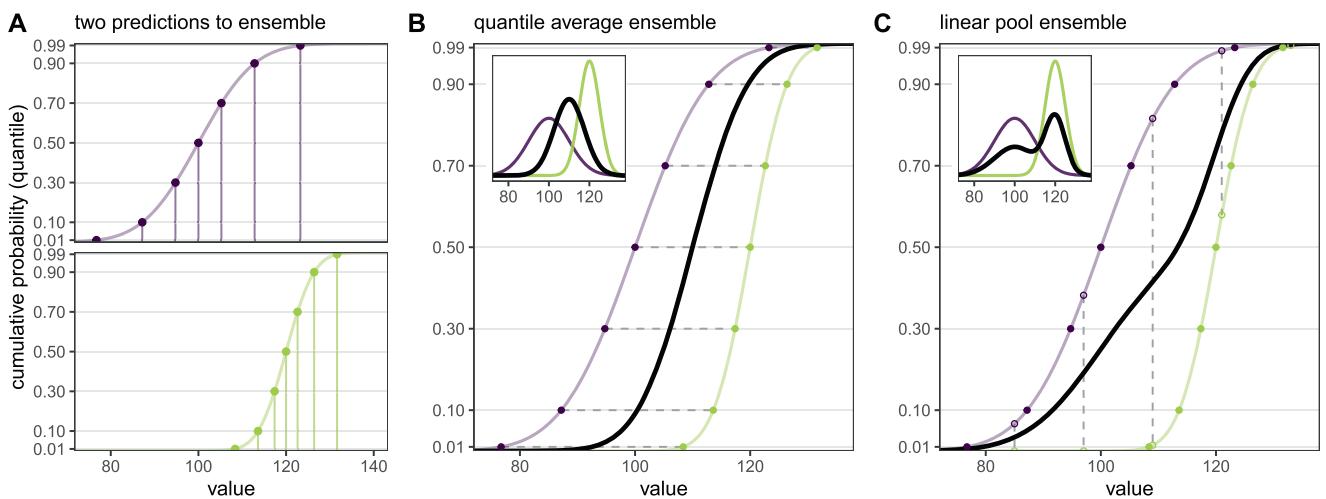


FIGURE 2 | (Panel A) Example predictions from two distributions ($N(100, 10)$ in purple and $N(120, 5)$ in green) shown as cumulative distribution functions (CDFs). To ease submission to a hub, the prediction can be summarized at a fixed number of points along the distribution. Here, the solid points show model output for seven fixed quantile levels ($\theta = 0.01, 0.1, 0.3, 0.5, 0.7, 0.9$, and 0.99). The y-axis ticks show each of the fixed quantile levels. The associated values for each fixed quantile level are shown with vertical lines. (Panel B) Quantile average ensemble, which is calculated by averaging values for each fixed quantile level (represented by horizontal dashed gray lines). The distributions and corresponding model outputs from panel A are re-plotted and the black line shows the resulting quantile average ensemble. Inset shows corresponding probability density functions (PDFs). (Panel C) Linear pool ensemble, which is calculated by averaging cumulative probabilities for each fixed value (represented by vertical dashed gray lines). The distributions and corresponding model outputs from panel A are re-plotted. To calculate the linear pool in this case, where model outputs are not defined for the same values (i.e., vertical lines in Panel A do not line up), the model outputs are used to interpolate the full CDF for each distribution from which predicted cumulative probabilities can be extracted for fixed values (shown with open circles). The black line shows the resulting linear pool average ensemble. Inset shows corresponding PDFs.

$\sum_{i=1}^N w_i \sigma_i^2 + \sum_{i=1}^N w_i (\mu_i - \mu_{LOP})^2$, where μ_i is the mean and σ_i^2 is the variance of individual prediction i , and although there is no closed-form variance for the quantile average, the variance of the quantile average will always be less than or equal to that of the linear pool [33]. Both methods generate distributions with the same mean, $\mu_Q = \mu_{LOP} = \sum_{i=1}^N w_i \mu_i$, which is the mean of individual model means [33]. The linear pool method preserves variation between individual models, whereas the quantile average cancels away this variation under the assumption it constitutes sampling error [24].

2.3 | Applications in Public Health and Infectious Disease Outbreaks

Multi-model ensembles have become the gold standard for forecasting and prediction efforts that support public health in real time [12, 34–36]. One prominent domain is forecasting key characteristics of infectious disease outbreaks, including weekly disease incidence or healthcare demand over future weeks, disease burden for the entire season, and timing and magnitude of the outbreak peak [14–16, 23, 37]. Projections of disease outcomes under multiple possible future scenarios have also been used to estimate intervention effectiveness to inform policy [38–40], and it has been proposed to use short-term forecasts of disease incidence to inform vaccine efficacy trials [41]. Standard guidelines for reporting of prediction efforts in outbreak and public health settings have also been established [42].

Across a variety of pathogens and outbreak settings, multi-model ensembles have been shown to produce forecasts that are as good or better than the individual models that compose the ensemble

[10, 13, 14, 16–19, 23]. Notably, the ensemble does not always outperform the best model, but typically offers improved consistency and robustness over individual models [16, 17, 20, 43]. However, in one instance, a baseline historical average of West Nile Virus cases in the US outperformed most model predictions, including the ensemble [15].

Examinations of ensemble methodology in infectious disease contexts suggest there is not one method that universally performs best. In short-term forecasting settings, a simple linear pool average of component predictions tends to produce prediction intervals that are too wide (i.e., suggesting outcomes are more uncertain than in reality); beta-transformation [44] and dynamic weighting [45] have been suggested to mitigate this problem. A median quantile average has been shown to provide similar performance to a weighted mean in short-term forecasting challenges, while also offering robustness to changes in performance across individual models [25], and was thus used as the primary ensemble for short-term forecasts of COVID-19 [16, 17]. For longer-term predictions of COVID-19, a trimmed LOP ensemble performed best, as models tended to be more overconfident in this setting [18]. The number of models submitting real-time predictions has varied dramatically (from as few as four models for longer-term predictions of COVID-19 [18] to more than 40 for short-term forecasts of COVID-19 [16]). Research including applications across influenza and COVID-19 suggests that at least 3 models are needed, with diminishing returns for every model that is added [46].

The growing body of literature on multi-model ensembles in public health domains emphasizes the utility of these approaches

to inform response in real time. Future research on optimizing ensemble performance for different targets and time horizons will further improve utility. Moreover, expanding the use of these methods to other pathogens and countries will enable further methodological development.

BOX 1 | Keywords and definitions: Concepts in statistical forecasting.

- *Ensemble*: the combination of outputs from multiple models into a single, aggregate prediction
- *Forecast*: a specific quantified prediction of an observable event or trend that has yet to be observed, conditional on data that has been observed up to a specified time.
- *Horizon*: time frame over which predictions are generated (e.g., 4 weeks ahead)
- *Linear opinion pool*: method for ensembling probabilistic predictions, which treats constituent probability distributions as alternative predictions of the future. Also called probability average.
- *Scenario projection*: estimates of future observations of future trends conditional on specific assumptions about a given scenario. Scenarios describe possible future conditions in terms of model parameters that might be varied, such as transmissibility, vaccine adoption, vaccine efficacy, the emergence of a new variant, etc.
- *Prediction*: estimate of current or future outcomes, typically generated from a model. Here, used to encapsulate a range of outcomes that could be modeled, encompassing short-term forecasts, scenario projections, and parameter estimates.
- *Point prediction*: a prediction represented by one single estimate, which generally represents a typical or common future outcome (e.g., a mean or median)
- *Probabilistic prediction*: a prediction that quantifies a probability distribution over future outcomes. This enables quantification of most likely outcomes and the risk of extreme events.
- *Model weighting*: a set of real numbers that defines the relative contribution of each constituent prediction in the ensemble (e.g., equal weighting or performance-based weighting).
- *Quantile average*: method for ensembling probabilistic predictions, which treats constituent probability distributions as uncertain estimates of a single true distribution. Also called Vincent average.
- *Quantile level/cumulative probability*: the y value in a cumulative distribution function, representing the cumulative probability of a future value (e.g., 50% cumulative probability)
- *Value of the target variable/quantile*: the x value in a cumulative distribution function, or the future values over which the probability distribution is defined (e.g., incident hospitalizations)

3 | How to Implement Ensemble Calculations

The methods described in Section 2 are implemented via the hubEnsembles package in a flexible, easy-to-use framework. Importantly, hubEnsembles is situated within the broader hubverse software infrastructure, which provides data standards and conventions for representing and working with model predictions [30], including for example, collecting and manipulating predictions (hubUtils) as well as visualization (hubVis). In 2024–2025, the hubverse supported over a dozen collaborative modeling hubs used by public health agencies across the globe. We begin with a short overview of hubverse concepts and conventions that support the process of combining model predictions, supplemented by example predictions provided by the hubverse in hubExamples, then explain the implementation of the two primary ensembling functions included in the package, simple_ensemble() and linear_pool(). Box 2 provides a glossary of key hubverse terms and definitions for reference.

3.1 | Terminology and Data Standards in the Hubverse

In the hubverse, predictions are always represented in a standardized tabular format called “model output”, codified by the model_out_tbl S3 class in hubUtils (a package of basic utility functions). Each row represents a single, unique prediction while the columns provide information about what is being predicted, its scope, and value. A single model output object can store and organize many predictions while remaining easy to parse at a glance, which is particularly useful when collecting predictions from multiple models to combine into an ensemble. Any tabular predictions can be transformed into model output using the as_model_out_tbl() function from hubUtils (see Section 5 for an example).

The model_out_tbl class is defined by four standard types of columns: (i) the model ID, which denotes which model has produced the prediction; (ii) the task IDs (also referred to as “task ID variables” or “task ID columns”), which provide details about what is being predicted; (iii) the model output representation, which specify the type of prediction and other identifying information; and (iv) the value of the prediction itself. While most of these columns are always required and have standardized column names, the task ID variables may vary according to the needs of the modeling hub or modeling task [30].

Table 1 provides an example of model output that stores short-term forecasts of weekly influenza hospitalizations for different US states and territories. By reading across the table, we can see that these are quantile predictions (output_type) of the quartiles (output type ID: otid) from a single model (model_id of “team1-mod”) for four distinct forecast horizons. Here, details about the prediction related to modeling task are represented by the task ID variables loc (location abbreviation), ref_date (reference date: the “starting point” of the forecasts), h (horizon: how many weeks into the future, relative to the ref_date), and target (what is being predicted).

As mentioned previously, task ID variables are not fixed in name, number, or composition to incorporate flexibility in the

TABLE 1 | Example of forecasts for incident influenza hospitalizations, formatted according to hubverse standards.

Model_id	Loc	ref_date	h	Target	Output_type	otid	Value
team1-mod	MA	2022-12-17	0	wk flu hosp	quantile	0.25	514
team1-mod	MA	2022-12-17	0	wk flu hosp	quantile	0.5	596
team1-mod	MA	2022-12-17	0	wk flu hosp	quantile	0.75	713
team1-mod	MA	2022-12-17	1	wk flu hosp	quantile	0.25	563
team1-mod	MA	2022-12-17	1	wk flu hosp	quantile	0.5	664
team1-mod	MA	2022-12-17	1	wk flu hosp	quantile	0.75	803
team1-mod	MA	2022-12-17	2	wk flu hosp	quantile	0.25	469
team1-mod	MA	2022-12-17	2	wk flu hosp	quantile	0.5	575
team1-mod	MA	2022-12-17	2	wk flu hosp	quantile	0.75	705
team1-mod	MA	2022-12-17	3	wk flu hosp	quantile	0.25	324
team1-mod	MA	2022-12-17	3	wk flu hosp	quantile	0.5	408
team1-mod	MA	2022-12-17	3	wk flu hosp	quantile	0.75	512

Note: Quantile predictions for the median and 50% prediction intervals from a single model are shown for four distinct horizons. The `output_type_id` column's name has been shortened to `otid` for brevity. These predictions are a modified subset of the `forecast_outputs` data provided by the `hubExamples` package.

`model_out_tbl` class. Different modeling efforts may use different sets of task ID columns with different values to define their prediction goals, or may simply choose distinct names to represent the same concept. For example, the date task column was named `ref_date` above but could easily be called `origin_date` or `forecast_date` instead. Some standard examples of task ID variables are available on the hubverse documentation website [30].

The “model output representation” columns `output_type` and `output_type_id` contain metadata about how the predictions are conveyed. The hubverse data standards require that these columns are included in a `model_out_tbl`. The `output_type` column defines how a prediction is represented and may be “mean” or “median” for point predictions, or one of “quantile”, “cdf”, “pmf”, or “sample” for probabilistic predictions. The `output_type_id` provides additional identifying information for prediction and is specific to the particular `output_type` (see Table 2). The last column, `value`, always contains the numeric value of the prediction, regardless of output type. Requirements for the values of the `output_type_id` and `value` columns associated with each valid output type are summarized in Table 2.

In addition to these six output types mentioned above, other types of predictions not explicitly defined in the hubverse may be represented within model output. For example, trajectories may be encoded using the “sample” output type while predictions of binary events can be captured using the “pmf” output type.

All output types can summarize predictions from univariate marginal distributions, for example, for a single location and time point. The sample output type—which represents randomly drawn values from a probabilistic predictive distribution—is unique in that it can additionally represent predictions from joint predictive distributions. This means that samples may encode dependence across combinations of multiple values for task ID variables, for example across multiple locations and/or time points. One common example is a trajectory that represents a single model run across all time points, and in such cases collecting predictions as samples can offer benefits [47]. In this case,

rows of sample predictions with the same index (specified by the `output_type_id`) from a particular model may be assumed to correspond to a single sample from a joint distribution.

For quantile predictions, the `output_type_id` is a numeric value between 0 and 1 specifying the cumulative probability associated with the quantile prediction. In the notation we defined in Section 2, the `output_type_id` corresponds to θ and the `value` is the quantile prediction $F^{-1}(\theta)$. For CDF or PMF predictions, the `output_type_id` is the target variable value x at which the cumulative distribution function or probability mass function for the predictive distribution should be evaluated, and the `value` column contains the predicted $F(x)$ or $f(x)$, respectively.

The hubverse also provides standards for target data (i.e., observed data corresponding to each prediction target), which can be stored in one of two formats: target time series or oracle output. The two tabular representations differ in terms of columns and purposes. Target time series data is a more traditional representation of the observed “truth” in a time series format with minimal columns; this format usually serves as calibration data for generating forecasts or might be used for visualization of predictions. Oracle output, on the other hand, represents prediction that an “oracle model” would have made had it known the observed values in advance. This format resembles model output data and is suited for evaluating forecasts. Some examples of target data are given in Section 4.

3.2 | Ensemble Functions in `hubEnsembles`

The `hubEnsembles` package contains two functions that perform ensemble calculations: `simple_ensemble()`, which applies some function to each model prediction, and `linear_pool()`, which computes an ensemble using the linear opinion pool method. In the following sections, we outline the implementation details for each function and how these implementations correspond to the statistical ensembling methods described in Section 2. A short description of the calculation performed by each function is summarized by output type in Table 3.

TABLE 2 | A table summarizing how the model output representation columns are used for predictions of different output types; adapted from hubverse documentation [30].

output_type	output_type_id	value
mean	NA (not used for mean predictions)	Numeric: The mean of the predictive distribution
median	NA (not used for median predictions)	Numeric: The median of the predictive distribution
quantile	Numeric between 0.0 and 1.0: A quantile level	Numeric: The quantile of the predictive distribution at the quantile level specified by the output_type_id
cdf	String or numeric naming a possible value of the target variable	Numeric between 0.0 and 1.0: The cumulative probability of the predictive distribution at the value of the outcome variable specified by the output_type_id
pmf	String naming a possible category of a discrete outcome variable	Numeric between 0.0 and 1.0: The probability mass of the predictive distribution when evaluated at a specified level of a categorical outcome variable
sample	Integer or string identifying the sample ^a	Numeric: A single sample value from the predictive distribution

^aRows of sample predictions from a particular model that share an output type ID value may be assumed to represent a single sample from a joint distribution across multiple levels of the task ID variables.

BOX 2 | Keywords and definitions: Terminology from the hubverse.

- **Modeling Task:** a definition of the goals of a modeling effort, possibly including conditions, assumptions, and targets (collectively known as task ID variables). Some tasks may be fixed across rounds, such as for forecast hubs that regularly solicit predictions for a set time horizon in the near-term future. Other tasks may be more variable; for example, those in scenario hubs that model hypothetical futures with different assumptions for different modeling rounds.
- **Compound modeling task:** a single, modeled unit with a multivariate outcome of interest, such as an epidemic trajectory across multiple weeks
- **Model output:** a dataset containing predictions of modeling tasks in tabular format generated in response to some modeling task for a specific round. A model might result from a single team's response to the task or from an ensemble of results representing the outcomes of multiple efforts.
- **Output type:** a classification of model output that is supported by the hubverse (i.e., the types of predictions that can be processed). See Table 2 for definitions of each.
- **Round:** a time period for which a set of specific model outputs is solicited. Rounds define the “cadence” of submission for a modeling hub. For example, some hubs might accept daily submissions, where each day is considered a different round. Other hubs might have one round every month, with a submission period that may be open for multiple days.
- **Task ID variables:** a collection of conditions, assumptions, and potentially targets that are used to parameterize a model task. These represent columns in the model output. A more detailed explanation of task ID variables can be found in the documentation.
- **Target:** an outcome of interest for a modeling hub (e.g., “incident case counts”). Targets typically (and sometimes implicitly) refer to a value of an observable variable in a given window of time, a given location, and possibly other stratifications (such as age group).

3.2.1 | Simple Ensemble

The `simple_ensemble()` function directly computes an ensemble from component model outputs by combining them via an aggregation function (C) within each unique combination of task ID variables, output types, and output type IDs. This function can be used to summarize predictions of output types mean, median, quantile, CDF, and PMF. Samples must be converted to a different output type (e.g., summarized into a CDF) in order to be ensembled using this function, as there is no meaningful equivalent for individual predictive draws. The mechanics of the ensemble calculations are the same for each of the output types, though the resulting statistical ensembling method differs for different output types (Table 3).

By default, `simple_ensemble()` uses the mean for the aggregation function C and equal weights for all models. For point predictions with a mean or median output type, the resulting ensemble prediction is an equally weighted average of the individual models' predictions. For probabilistic predictions in a quantile format, by default `simple_ensemble()` calculates an equally weighted average of individual model target variable values at each quantile level, which is equivalent to a quantile average. For model outputs in a CDF or PMF format, by default `simple_ensemble()` computes an equally weighted average of individual model (cumulative or bin) probabilities at each target variable value, which is equivalent to the linear pool method.

Any aggregation function C may be specified by the user. For example, a median ensemble may also be created by specifying `median` as the aggregation function, or a custom function may be passed to the `agg_fun` argument to create other ensemble types. Similarly, model weights can be specified to create a weighted ensemble.

3.2.2 | Linear Pool

The `linear_pool()` function implements the linear opinion pool (LOP) method for ensembling predictions. This function can be used to combine predictions with output types mean, quantile, CDF, PMF, and sample. Since the LOP is an average of probabilities, it does not support ensembles of medians. Unlike `simple_ensemble()`, this function handles its computation

TABLE 3 | Summary of ensemble function calculations for each output type.

output_type	simple_ensemble(..., agg_fun=mean)	linear_pool()
mean	Mean of individual model means	Mean of individual model means
median	Mean of individual model medians	NA
quantile	Mean of individual model target variable values at each quantile level, $F_Q^{-1}(\theta)$	Quantiles of the distribution are obtained by computing the mean of estimated individual model cumulative probabilities at each target variable value, $F_{LOP}^{-1}(\theta)$
cdf	Mean of individual model cumulative probabilities at each target variable value, $F_{LOP}(x)$	Mean of individual model cumulative probabilities at each target variable value, $F_{LOP}(x)$
pmf	Mean of individual model bin or category probabilities at each target variable value, $f_{LOP}(x)$	Mean of individual model bin or category probabilities at each target variable value, $f_{LOP}(x)$
sample	NA	Samples of the distribution are obtained by stratified draw from individual models' samples

Note: The ensemble function determines the operation that is performed, and in the case of probabilistic output types (quantile, CDF, PMF), this also determines what ensemble distribution is generated (quantile average, $F_Q^{-1}(\theta)$, or linear pool, $F_{LOP}(x)$). The ensembled predictions are returned in the same output type as the inputs. Thus, the output type determines how the resulting ensemble distribution is summarized (as a quantile, $F^{-1}(\theta)$, cumulative probability, $F(x)$, or probability $f(x)$). Estimating individual model cumulative probabilities is required to compute a `linear_pool()` for predictions of quantile output type; see Section 3.2.2 on the linear pool operation for details. In the case of `simple_ensemble()`, we show the calculations for the default case where `agg_fun = mean`; however, if another aggregation function is chosen (e.g., `agg_fun = median`), that calculation would be performed instead. For example, `simple_ensemble(..., agg_fun = median)` applied to predictions of mean output type would return the median of individual model means.

differently based on the output type. For the CDF, PMF, and mean output types, the linear pool method is equivalent to calling `simple_ensemble()` with a mean aggregation function (see Table 3), since `simple_ensemble()` produces a linear pool prediction (an average of individual model cumulative or bin probabilities).

3.2.2.1 | Linear Pool of samples. For the sample output type, the LOP method collects a stratified draw of the individual models' predictions and pools them into a single ensemble distribution. By default, all samples are used to create this ensemble. Additionally, only equally-weighted linear pools of samples are supported by the `hubEnsembles` package during this time. Samples may also be converted to another common output type such as quantiles or bin probabilities (as the main scientific interest often concerns a summary of samples), and other ensemble methods may then be utilized for that output type.

When requesting a subsetted ensemble of samples, it becomes important to distinguish between marginal and joint predictive distributions so that the dependence structure can be passed to `linear_pool()`. We introduce the concepts of the compound task ID set and derived task ID variables here, as they help identify the underpinnings the dependence structure of the ensemble predictive distributions.

As stated in the previous section, the sample output type is unique in that it can represent predictions from both marginal distributions and joint distributions. Hence, samples can encode dependence across combinations of multiple values for task ID variables, for example, across multiple locations and/or time points. In this case, sample predictions with the same index (output type ID) from a given model may be assumed to correspond to a single trajectory.

The compound task id set consists of independent task id variables that, together, identify a "compound modeling task" corresponding to a single, modeled unit with a multivariate outcome of

interest. Samples summarizing a marginal distribution will generally have a compound task ID set composed of all the task ID variables (except for derived task IDs). Conversely, samples summarizing a joint distribution will have a compound task ID set only containing task ID variables for which the joint distribution does not capture dependencies. For example, if a joint distribution is estimated across multiple forecast horizons separately for each location (i.e., horizon-based trajectories), the task ID "location" would be included in the compound task ID set but "horizon" would not.

Derived task IDs are another subgroup of task ID variables that must be specified in a call to `linear_pool()` for a subsetted sample ensemble; their values are derived from a combination of the values from other task ID variables (which may or may not be part of the compound task ID set). A common example of a derived task ID variable is the target date for a prediction, which is a deterministic function of the reference date of the prediction and the prediction horizon. Generally, the derived task IDs won't be included in the compound task ID set because they are not needed to identify a single modeled unit for an outcome of interest, *unless* all of the task ID variables their values depend on are already a part of the compound task ID set.

Not all model outputs will contain derived task IDs, in which case the argument may be set to `NULL` (the default value). However, it is important to provide the `linear_pool()` function with any derived task IDs when calculating an ensemble of (subsetted) samples, as they are used to check that the provided compound task ID set is compatible with the input predictions and the resulting LOP is valid.

3.2.2.2 | Linear Pool of quantiles. For the quantile output type, calculation of LOP requires a few extra steps. This is because LOP averages cumulative probabilities at each value of the target variable, but the predictions are given as quantiles (on the scale of the target variable) for fixed quantile levels. The value for these quantile predictions will generally differ between models; hence,

we are typically not provided cumulative probabilities at the same values of the target variable for all component predictions. This lack of alignment between cumulative probabilities for the same target variable values impedes computation of LOP from quantile predictions and is illustrated in panel A of Figure 2.

Given that LOP cannot be directly calculated from quantile predictions, we must first obtain an estimate of the CDF for each component distribution from the provided quantiles, combine the CDFs, then calculate the quantiles using the ensemble's CDF. We perform this calculation in three main steps, assisted by the `distfromq` package [48] for the first two:

1. Interpolate and extrapolate from the provided quantiles for each component model to obtain an estimate of the CDF of that particular distribution.
2. Draw samples from each component model distribution. To reduce Monte Carlo variability, we use quasi-random samples corresponding to quantiles of the estimated distribution [49].
3. Pool the samples from all component models and extract the desired quantiles.

For step 1, functionality in the `distfromq` package uses a monotonic cubic spline for interpolation on the interior of the provided quantiles. The user may choose one of several distributions to perform extrapolation of the CDF tails. These include normal, lognormal, and Cauchy distributions, with “normal” set as the default. A location-scale parameterization is used, with separate location and scale parameters chosen in the lower and upper tails so as to match the two most extreme quantiles. The sampling process described in steps 2 and 3 approximates the linear pool calculation described in Section 2.

4 | A Simple Demonstration of Multi-Model Ensembles

In this section, we provide a simple example to illustrate how to compute a multi-model ensemble and compare the methods

supported by the functions of `hubEnsembles`. In doing so, we use a number of other packages available through the hubverse, including to access example data and to visualize outputs. See the Data Availability Statement for details about implementation and required package versions.

4.1 | Example Data: A Forecast Hub

The first step in generating a multi-model ensemble is to gather the predictions we wish to combine. In this example, we use some short-term forecasts already formatted as model output data from the `hubExamples` package. These model outputs are from a larger example modeling hub, created using a modified subset of predictions from the FluSight Forecasting challenge (discussed in further detail in Section 5). In addition to toy model output data, the example hub also includes observed data in the form of target time series data and oracle output.

The model output is stored in a data object named `forecast_outputs` and contains mean, median, quantile, and sample forecasts of future incident influenza hospitalizations, as well as CDF and PMF forecasts of hospitalization intensity. Each prediction is described by five task ID variables: the location for which the forecast was made (`location`), the date on which the forecast was made (`reference_date`), the number of steps ahead (`horizon`), the date of the forecast prediction (`target_end_date`, a combination of the date the forecast was made and the forecast horizon), and the forecast target (`target`). We begin by examining the predictions for weekly incident influenza hospitalizations, displaying a subset of each output type in Table 4.

While the `hubExamples` package provides both formats of target data, we focus on the target time series data (Table 5) which is convenient for making forecasts and plotting. The `forecast_target_ts` data object provides observed values for weekly incident influenza hospitalizations and weekly rate change of influenza hospitalizations in a given week

TABLE 4 | Example model output for forecasts of weekly incident influenza hospitalizations.

model_id	target	horizon	output_type	output_type_id	value
Flusight-baseline	wk inc flu hosp	1	mean	NA	582.07
Flusight-baseline	wk inc flu hosp	1	median	NA	582.00
Flusight-baseline	wk inc flu hosp	1	quantile	0.05	496.00
Flusight-baseline	wk inc flu hosp	1	quantile	0.25	566.00
Flusight-baseline	wk inc flu hosp	1	quantile	0.75	598.00
Flusight-baseline	wk inc flu hosp	1	quantile	0.95	668.00
Flusight-baseline	wk inc flu hosp	1	sample	2101	606.00
Flusight-baseline	wk inc flu hosp	1	sample	2102	576.00
Flusight-baseline	wk inc flu hosp	1	sample	2103	578.00
MOBS-GLEAM_FLUH	wk inc flu hosp	1	mean	NA	704.73
MOBS-GLEAM_FLUH	wk inc flu hosp	1	median	NA	664.00
MOBS-GLEAM_FLUH	wk inc flu hosp	1	quantile	0.05	446.00

Note: A subset of example model output is shown: 1-week ahead forecasts made on 2022-12-17 for Massachusetts from three distinct models; only the mean, median, select samples, and the 5th, 25th, 75th, and 95th quantiles are displayed. The `location`, `reference_date` and `target_end_date` columns have been omitted for brevity. This example data is provided in the `hubExamples` package.

TABLE 5 | Example target time series data for incident influenza hospitalizations and (weekly) influenza hospitalization rate change.

target_end_date	target	location	observation
2022-11-05	wk inc flu hosp	25	31.000
2022-11-12	wk inc flu hosp	25	43.000
2022-11-19	wk inc flu hosp	25	79.000
2022-11-26	wk inc flu hosp	25	221.000
2022-12-03	wk inc flu hosp	25	446.000
2022-12-10	wk inc flu hosp	25	578.000
2022-12-17	wk inc flu hosp	25	694.000
2022-12-24	wk inc flu hosp	25	769.000
2022-12-31	wk inc flu hosp	25	733.000
2023-01-07	wk inc flu hosp	25	466.000
2023-01-14	wk inc flu hosp	25	238.000
2023-01-21	wk inc flu hosp	25	122.000
2023-01-28	wk inc flu hosp	25	71.000
2022-11-05	wk flu hosp rate	25	0.444
2022-11-12	wk flu hosp rate	25	0.616
2022-11-19	wk flu hosp rate	25	1.132
2022-11-26	wk flu hosp rate	25	3.167
2022-12-03	wk flu hosp rate	25	6.391
2022-12-10	wk flu hosp rate	25	8.282
2022-12-17	wk flu hosp rate	25	9.945
2022-12-24	wk flu hosp rate	25	11.019
2022-12-31	wk flu hosp rate	25	10.503
2023-01-07	wk flu hosp rate	25	6.677
2023-01-14	wk flu hosp rate	25	3.410
2023-01-21	wk flu hosp rate	25	1.748
2023-01-28	wk flu hosp rate	25	1.017

Note: We round the latter target's observed values (in the observation column) to three digits for readability. This table displays target time series data for Massachusetts (FIPS code 25) between 2022-11-01 and 2023-02-01. The target data is provided in the hubExamples package.

for a particular location using columns `observation`, `target_end_date`, `location`, and `target`. The forecast-specific task ID variables `reference_date` and `horizon` are not relevant for this time series representation of the target data, and are thus not included as columns.

We can plot the quantile and median forecasts and the target time series data (Figure 3) shown above using the `plot_step_ahead_model_output()` function from hubVis, another package in the hubverse suite for visualizing model outputs. We subset the model output data and the target data to the target, location, and time horizons we are interested in.

```
> model_outputs_plot <- hubExamples::forecast_outputs |>
+ hubUtils::as_model_out_tbl() |>
+ dplyr::filter(
+   location == "25",
+   output_type %in% c("median", "quantile"),
+   reference_date == "2022-12-17"
+ ) |>
+ dplyr::mutate(output_type_id = as.double(output_type_id))
> target_data_plot <- hubExamples::forecast_target_ts |>
+ dplyr::filter(
+   target == "wk inc flu hosp",
+   location == "25",
+   target_end_date >= "2022-11-01", target_end_date <=
+   "2023-03-01"
+ )
```

```
> hubVis::plot_step_ahead_model_output(
+   model_out_tbl = model_outputs_plot,
+   target_data = target_data_plot,
+   facet = "model_id",
+   facet_nrow = 1,
+   interactive = FALSE,
+   intervals = c(0.5, 0.9),
+   show_legend = FALSE,
+   use_median_as_point = TRUE,
+   x_target_col_name = "target_end_date",
+   x_col_name = "target_end_date"
+ ) +
+   theme_bw() +
+   labs(y = "incident hospitalizations")
```

Some hubs request modeling teams submit samples instead of, or in addition to, predicted distributions (Figure 4). Samples can be useful for capturing additional correlations across targets that are difficult to represent when predictions are summarized using quantile or CDF output types. In this example, separate probabilistic predictions of incident influenza hospitalizations in each week cannot capture the shape of possible epidemic trajectories, but multivariate samples for the trajectory of disease incidence across weeks can. We filter to the sample output type and plot. These samples are representative draws from the distributions plotted in Figure 3.

Next, we examine the PMF forecasts for hospitalization intensity in the example model output data. For this target, teams forecasted the probability that hospitalization intensity will be “low”, “moderate”, “high”, or “very high”. These four categories are determined by thresholds for weekly hospital admissions per 100,000 population. In other words, “low” hospitalization intensity in a given week means few incident influenza hospitalizations per 100,000 population are predicted, whereas “very high” hospitalization intensity means many hospitalizations per 100,000 population are predicted. These forecasts are made for the same task ID variables as the `quantile` forecasts of incident hospitalizations except for the target, which is “wk flu hosp rate category” for these categorical predictions.

We show a representative example of the hospitalization intensity category forecasts in Table 6. Because these forecasts are PMF output type, the `output_type_id` column specifies the bin of hospitalization intensity and the `value` column provides the forecasted probability of hospitalization incidence being in that category. Values sum to 1 across bins. For the MOBS-GLEAM_FLUH and PSI-DICE models, incidence is forecasted to decrease over the horizon (Figure 3), and correspondingly, there is lower probability of “high” and “very high” hospitalization intensity for later horizons (Figure 5).

4.2 | Creating Ensembles With `simple_ensemble()`

Using the default options for `simple_ensemble()`, we can generate an equally weighted mean ensemble for each unique combination of values for the task ID variables, the `output_type` and the `output_type_id`. Recall that this function corresponds to different statistical ensemble methods for different output types: for the quantile output type in our example data, the resulting ensemble is a quantile average, while for the CDF and PMF output types, the ensemble is a linear pool (Table 3).

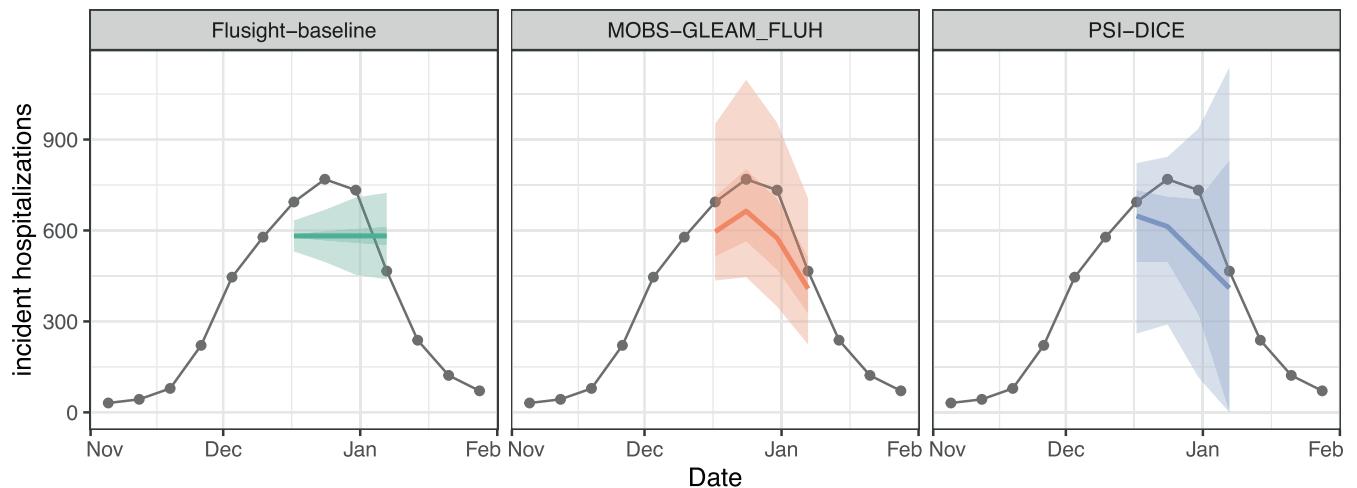


FIGURE 3 | One example set of quantile forecasts for weekly incident influenza hospitalizations in Massachusetts from each of three models (panels). Forecasts are represented by a median (line), 50% and 90% prediction intervals (ribbons). Gray points represent observed incident hospitalizations.

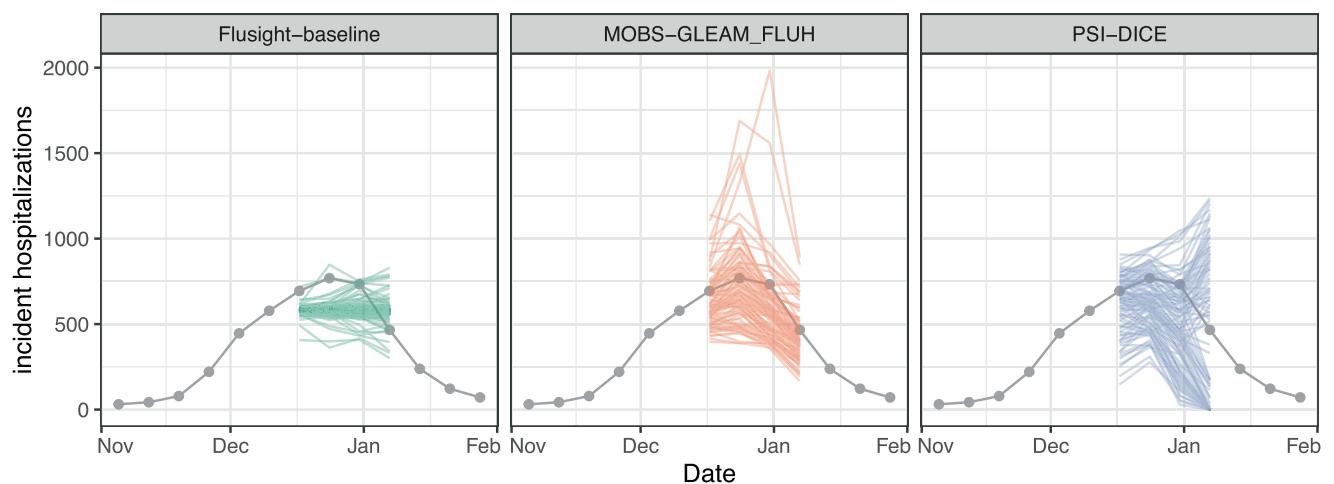


FIGURE 4 | One example set of sample forecasts for weekly incident influenza hospitalizations in Massachusetts from each of three models (panels). Each sample is represented by a single line. Gray points represent observed incident hospitalizations.

TABLE 6 | Example PMF model output for forecasts of incident influenza hospitalization intensity.

model_id	target	horizon	output_type	output_type_id	value
Flusight-baseline	wk flu hosp rate category	1	pmf	low	0.000
Flusight-baseline	wk flu hosp rate category	1	pmf	moderate	0.003
Flusight-baseline	wk flu hosp rate category	1	pmf	high	0.073
Flusight-baseline	wk flu hosp rate category	1	pmf	very high	0.924
MOBS-GLEAM_FLUH	wk flu hosp rate category	1	pmf	low	0.000
MOBS-GLEAM_FLUH	wk flu hosp rate category	1	pmf	moderate	0.002
MOBS-GLEAM_FLUH	wk flu hosp rate category	1	pmf	high	0.163
MOBS-GLEAM_FLUH	wk flu hosp rate category	1	pmf	very high	0.835
PSI-DICE	wk flu hosp rate category	1	pmf	low	0.013
PSI-DICE	wk flu hosp rate category	1	pmf	moderate	0.065
PSI-DICE	wk flu hosp rate category	1	pmf	high	0.218
PSI-DICE	wk flu hosp rate category	1	pmf	very high	0.704

Note: A subset of predictions are shown: 1-week ahead PMF forecasts made on 2022-12-17 for Massachusetts from three distinct models. We round the forecasted probability (in the value column) to two digits. The location, reference_date and target_end_date columns have been omitted for brevity. This example data is provided in the hubExamples package.

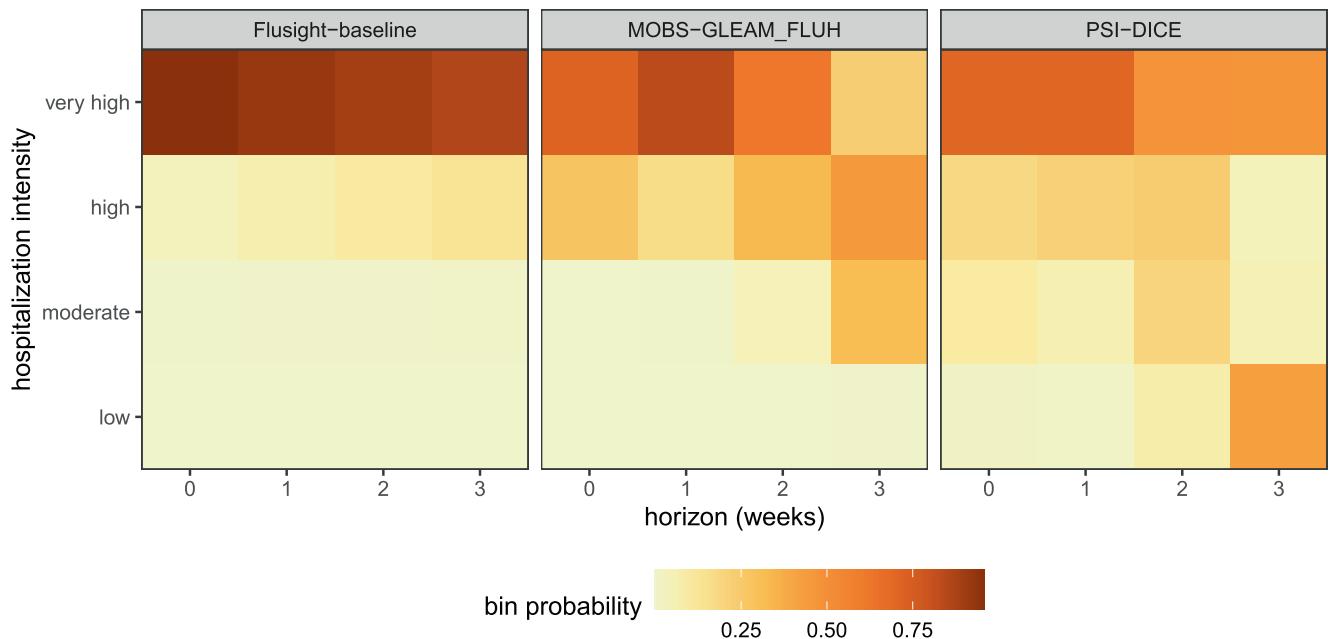


FIGURE 5 | One example PMF forecast of incident influenza hospitalization intensity is shown for each of three models (panels). Each cell shows the forecasted probability of a given hospitalization intensity bin (low, moderate, high, and very high) for each forecast horizon (0–3 weeks ahead). Darker colors indicate higher forecasted probability.

```
> mean_ens <- hubExamples::forecast_outputs |>
+   dplyr::filter(output_type != "sample") |>
+   hubEnsembles::simple_ensemble(
+     model_id = "simple-ensemble-mean"
+   )
```

The resulting model output has the same structure as the original model output data (Table 7), with columns for model ID, task ID variables, output type, output type ID, and value. We also use `model_id = "simple-ensemble-mean"` to change the name of this ensemble in the resulting model output; if not specified, the default is “hub-ensemble”.

4.2.1 | Changing the Aggregation Function

We can change the function that is used to aggregate model outputs. For example, we may want to calculate a median of the component models’ submitted values for each quantile. We do so by specifying `agg_fun = median`.

```
> median_ens <- hubExamples::forecast_outputs |>
+   dplyr::filter(output_type != "sample") |>
+   hubEnsembles::simple_ensemble(
+     agg_fun = median,
+     model_id = "simple-ensemble-median"
+   )
```

Custom functions can also be passed into the `agg_fun` argument. We illustrate this by defining a custom function to compute the ensemble prediction as a geometric mean of the component model predictions. Any custom function to be used must have an argument `x` for the vector of numeric values to summarize, and if relevant, an argument `w` of numeric weights.

```
> geometric_mean <- function(x) {
+   n <- length(x)
+   prod(x)^(1 / n)
+ }
> geometric_mean_ens <- hubExamples::forecast_outputs |>
+   dplyr::filter(output_type != "sample") |>
+   hubEnsembles::simple_ensemble(
+     agg_fun = geometric_mean,
+     model_id = "simple-ensemble-geometric"
+   )
```

As expected, the mean, median, and geometric mean each give us slightly different resulting ensembles. The median point estimates, 50% prediction intervals, and 90% prediction intervals in Figure 6 demonstrate this.

4.2.2 | Weighting Model Contributions

We can weight the contributions of each model in the ensemble using the `weights` argument of `simple_ensemble()`. This argument takes a `data.frame` that should include a `model_id` column containing each unique model ID and a `weight` column. In the following example, we include the baseline model in the ensemble, but give it less weight than the other forecasts.

```
> model_weights <- data.frame(
+   model_id = c("MOBS-GLEAM_FLUH", "PSI-DICE",
+   "Flusight-baseline"),
+   weight = c(0.4, 0.4, 0.2)
+ )
> weighted_mean_ens <- hubExamples::forecast_outputs |>
+   dplyr::filter(output_type != "sample") |>
+   hubEnsembles::simple_ensemble(
+     weights = model_weights,
+     model_id = "simple-ensemble-weighted-mean"
+   )
```

TABLE 7 | Mean ensemble model output.

model_id	target	horizon	output_type	output_type_id	value
simple-ensemble-mean	wk flu hosp rate category	1	pmf	high	0.15
simple-ensemble-mean	wk flu hosp rate category	1	pmf	low	0.00
simple-ensemble-mean	wk flu hosp rate category	1	pmf	moderate	0.02
simple-ensemble-mean	wk flu hosp rate category	1	pmf	very high	0.82
simple-ensemble-mean	wk inc flu hosp	1	mean	NA	627.09
simple-ensemble-mean	wk inc flu hosp	1	median	NA	619.67
simple-ensemble-mean	wk inc flu hosp	1	quantile	0.25	541.67
simple-ensemble-mean	wk inc flu hosp	1	quantile	0.75	704.33

Note: The values in the model_id column are set by the argument `simple_ensemble(..., model_id)`. Results are generated for all output types, but only a subset are shown: 1-week ahead forecasts made on 2022-12-17 for Massachusetts, with only the mean, median, 25th and 75th quantiles for the quantile output type and all bins for the PMF output type. The location, reference_date and target_end_date columns have been omitted for brevity, and the value column is rounded to two digits.

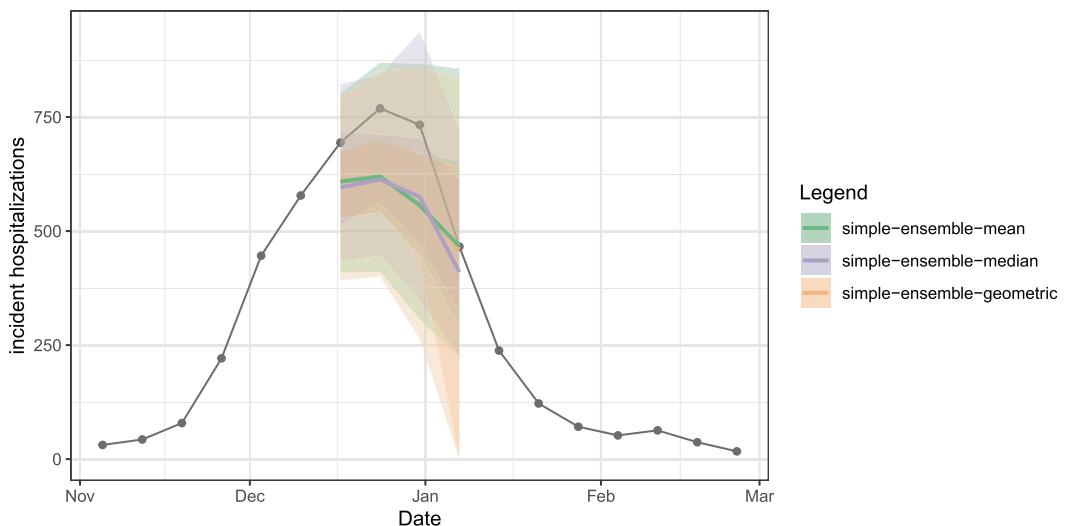


FIGURE 6 | Three different ensembles for weekly incident influenza hospitalizations in Massachusetts. Each ensemble combines individual predictions from the example hub (Figure 3) using a different method: arithmetic mean, geometric mean, or median. All methods correspond to variations of the quantile average approach. Ensembles are represented by a median (line), 50% and 90% prediction intervals (ribbons). Geometric mean ensemble and simple mean ensemble generate similar estimates in this case.

4.3 | Creating Ensembles With `linear_pool()`

We can also generate a linear pool ensemble, or distributional mixture, using the `linear_pool()` function; this function can be applied to predictions with an `output_type` of mean, quantile, sample, CDF, or PMF. Our example hub includes the median output type, so we exclude it from the calculation.

```
> linear_pool_ens <- hubExamples::forecast_outputs |>
+   dplyr::filter(output_type != "median") |>
+   hubEnsembles::linear_pool(model_id = "linear-pool")
```

As described above, for quantile model outputs, the `linear_pool` function approximates the full probability distribution for each component prediction using the value-quantile pairs provided by that model, and then obtains quasi-random samples from that distributional estimate. The number of samples drawn from the distribution of each component model defaults to `1e4`, but this can be changed using the `n_samples` argument.

In Figure 7, we compare ensemble results generated by `simple_ensemble()` and `linear_pool()` for model outputs of output types PMF and quantile. As expected, the results from the two functions are equivalent for the PMF output type: for this output type, the `simple_ensemble()` method averages the predicted probability of each category across the component models, which is the definition of the linear pool ensemble method. This is not the case for the quantile output type, because the `simple_ensemble()` is computing a quantile average.

4.3.1 | Weighting Model Contributions

Like with `simple_ensemble()`, we can change the default function settings. For example, weights that determine a model's contribution to the resulting ensemble may be provided. (Note that we must exclude the sample output type here because it cannot yet be combined into weighted ensembles.)

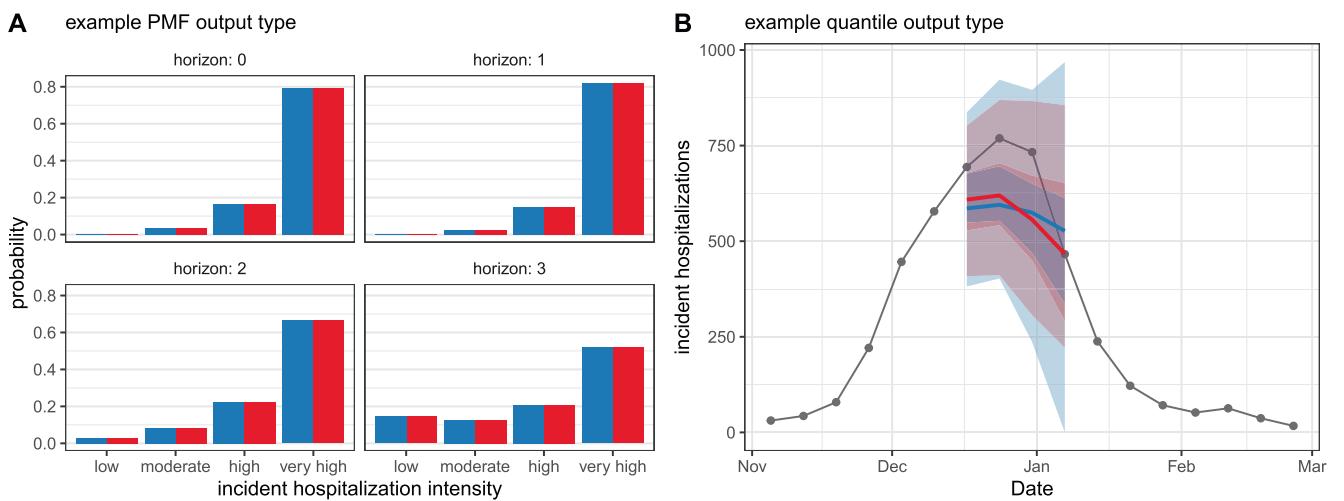


FIGURE 7 | Comparison of results from `linear_pool()` (blue) and `simple_ensemble()` (red). (Panel A) Ensemble predictions of Massachusetts incident influenza hospitalization intensity (classified as low, moderate, high, or very high), which provide an example of PMF output type. (Panel B) Ensemble predictions of weekly incident influenza hospitalizations in Massachusetts, which provide an example of quantile output type. Note, for quantile output type, `simple_ensemble()` corresponds to a quantile average. Ensembles combine individual models from the example hub, and are represented by a median (line), 50% and 90% prediction intervals (ribbons) (Figure 3).

```
> weighted_linear_pool_norm <- hubExamples::forecast_outputs |>
+   dplyr::filter(!output_type %in% c("median", "sample")) |>
+   hubEnsembles::linear_pool(
+     weights = model_weights,
+     model_id = "linear-pool-weighted"
+   )
```

4.3.2 | Changing the Parametric Family Used for Extrapolation Into Distribution Tails

When requesting a linear pool of quantiles, we may also change the distribution that `distfromq` uses to approximate the tails of component models' predictive distributions to either log normal or Cauchy using the `tail_dist` argument (the default is normal) [48]. This choice usually does not have a large impact on the resulting ensemble distribution, though, and can only be seen in its outer edges. (For more details and other function options, see the documentation in the `distfromq` package at <https://reichlab.io/distfromq/>.)

```
> linear_pool_lnorm <- hubExamples::forecast_outputs |>
+   dplyr::filter(output_type == "quantile") |>
+   hubEnsembles::linear_pool(
+     model_id = "linear-pool-lognormal",
+     tail_dist = "lnorm"
+   )
> linear_pool_cauchy <- hubExamples::forecast_outputs |>
+   dplyr::filter(output_type == "quantile") |>
+   hubEnsembles::linear_pool(
+     model_id = "linear-pool-cauchy",
+     tail_dist = "cauchy"
+   )
```

4.3.3 | Requesting a Subset of Input Sample Predictions to be Ensembled

Recall that for the sample output type, `linear_pool()` simply collects and pools the input sample predictions into a single distribution. By default, the resulting ensemble model output

contains all provided sample predictions so that the total number of samples for the ensemble is equal to the sum of the number of samples from all individual models.

```
> joint_lp_default <- hubExamples::forecast_outputs |>
+   dplyr::filter(output_type == "sample") |>
+   hubEnsembles::linear_pool(
+     weights = NULL,
+     model_id = "linear-pool-joint-subset",
+     task_id_cols =
+       c("reference_date", "location", "horizon",
+       "target",
+       "target_end_date")
+   )
```

To change this behavior, the user may instead specify a number of sample predictions for the ensemble to return using the `n_output_samples` argument. Then, a random subset of predictions from individual models will be selected to construct a LOP of samples so that all component models are represented equally. This random selection of samples is stratified by model to ensure approximately the same number of samples from each individual model is included in the ensemble.

The compound task ID set and any derived task IDs must also be specified to ensure the function returns a valid ensemble. For example, one compound task we are predicting for in the model output is the number of weekly incident influenza hospitalizations ("target") in Massachusetts ("location") starting on December 5, 2022 ("reference_date"). Here, "horizon" is not part of the compound task ID set, indicating that sample predictions made at each horizon depend on those for the other horizons within every compound task for the sample output type. Each sample can therefore be interpreted as a trajectory giving a possible path of hospitalizations over time. These three task id variables ("reference_date", "location", and "target") make up the compound task ID set that is specified in the call to `linear_pool()`. The remaining task ID variable "target_end_date" is a derived task ID since its value is a function of "reference_date" and "horizon".

```

> joint_lp_subset <- hubExamples::forecast_outputs |>
+   dplyr::filter(output_type == "sample") |>
+   hubEnsembles::linear_pool(
+     weights = NULL,
+     model_id = "linear-pool-joint-subset",
+     task_id_cols =
+       c("reference_date", "location", "horizon",
+       "target",
+       "target_end_date"),
+     compound_taskid_set = c("reference_date",
+     "location",
+     "target"),
+     derived_task_ids = "target_end_date",
+     n_output_samples = 100
+   )

```

5 | Example: In-Depth Analysis of Forecast Data

To further demonstrate the differences between the two ensemble functions and the utility of the `hubEnsembles` package, we provide a more complex example that walks through the full process of generating multi-model ensembles. This case study gathers real forecasts collected by a modeling hub to create four equally-weighted ensembles, then evaluates their performance to determine the best approach for the application.

The predictions we use to create the ensemble models are sourced from two seasons of the FluSight forecasting challenge. Since 2013, the US Centers for Disease Control and Prevention (CDC) has been soliciting short-term forecasts of seasonal influenza from modeling teams through this collaborative challenge [50]. Using `simple_ensemble()` and `linear_pool()`, we build four equally-weighted, multi-model ensembles to predict weekly influenza hospitalizations: a quantile (arithmetic) mean, a quantile median, a linear pool with normal tails, and a linear pool with lognormal tails. Then, we compare the resulting ensembles' performance through plotting and scoring their forecasts.

Only a select portion of the code used in this analysis is shown for brevity, but all the functions and scripts used to generate the case study results can be found in the associated GitHub repository (<https://github.com/hubverse-org/hubEnsemblesManuscript>). In particular, the figures and tables supporting this analysis can be reproduced using data from `.rds` files stored in the `analysis/data/raw-data` directory and scripts in the `inst` directory of the repository.

5.1 | Data and Methods

We collect the predictions used to generate the four ensembles by querying them from Zoltar [51], a repository designed to archive forecasts created by the Reich Lab at UMass Amherst. For this analysis we only consider FluSight forecasts in a quantile format from the 2021–2022 and 2022–2023 influenza seasons. These quantile forecasts are stored in two data objects, split by season, called `flu_forecasts-zoltar_21-22.rds` and `flu_forecasts-zoltar_22-23.rds`, which are then joined together into a single data frame. A subset is shown below in Table 8.

```

> flu_forecasts_raw_21_22 <- readr::read_rds(
+   here::here("analysis/data/raw_data/flu_forecasts-
+   zoltar_21-22.rds")
+ )
> flu_forecasts_raw_22_23 <- readr::read_rds(
+   here::here("analysis/data/raw_data/flu_forecasts-
+   zoltar_22-23.rds")
+ )
> flu_forecasts_raw <- rbind(flu_forecasts_raw_21_22,
+   flu_forecasts_raw_22_23)

```

Although these forecasts are in a tabular format, they are not `model_out_tbl` objects and thus cannot yet be fed into either of the `hubEnsembles` functions. Thus, we must use the `as_model_out_tbl()`¹ function from `hubUtils` to transform the raw forecasts so that they conform to hubverse standards. Below, we specify the appropriate column mappings in the call with task ID variables of `forecast_date` (when the forecast was made), `location`, `horizon`, and `target`.

```

> flu_forecasts_hubverse <- flu_forecasts_raw |>
+   dplyr::rename(forecast_date = timezero, location =
+     unit) |>
+   tidyverse::separate(target, sep = " ", convert = TRUE,
+     into = c("horizon", "target"),
+     extra = "merge")
|>
+   dplyr::mutate(target_end_date =
+     round_date(forecast_date + weeks(horizon),
+     "weeks") -
+     days(1)) |>
+   as_model_out_tbl(
+     model_id_col = "model",
+     output_type_col = "class",
+     output_type_id_col = "quantile",
+     value_col = "value",
+     sep = "-",
+     trim_to_task_ids = FALSE,
+     hub_con = NULL,
+     task_id_cols =
+       c("forecast_date", "location", "horizon", "target",
+       "target_end_date"),
+     remove_empty = TRUE
+   )

```

To ensure the quantile mean and median ensemble had consistent component forecast make-up at every quantile level, we only included predictions (defined by a unique combination of task ID variables) that contained all 23 quantiles specified by FluSight ($\theta \in \{.010, 0.025, .050, .100, \dots, .900, .950, .990\}$). This requirement required no further action on our part, since it was consistent with FluSight submission guidelines. However, we did remove the baseline and median ensemble models generated by the FluSight hub from the component forecasts, a choice motivated by the desire to match the composition of models in the official FluSight ensemble. (Note, though, that the component models included in our ensembles did not always exactly match those included in the FluSight-ensemble for every forecast week as a result of slightly different inclusion criteria.)

With these inclusion criteria, the final data set of component forecasts consists of predictions from 25 modeling teams and 42 distinct models, 53 forecast dates (one per week), 54 US locations, 4 horizons, 1 target, and 23 quantiles. In the 2021–2022 season, 25 models made predictions for 22 weeks spanning from late January 2022 to late June 2022, and in the 2022–2023 season, there were 31 models making predictions for 31 weeks spanning mid-October 2022 to mid-May 2023. Fourteen of the 42 total models made forecasts for both seasons. Locations consist of the 50 US states, Washington DC, Puerto Rico, the Virgin Islands, and the entire US; horizons 1 to 4 weeks ahead, quantiles the 23 described

TABLE 8 | An example prediction of weekly incident influenza hospitalizations pulled directly from Zoltar.

model	target	class	value	cat	prob	sample	quantile	family
UMass-trends_ensemble	1 wk ahead inc flu hosp	quantile	12	NA	NA	NA	0.025	NA
UMass-trends_ensemble	1 wk ahead inc flu hosp	quantile	17	NA	NA	NA	0.100	NA
UMass-trends_ensemble	1 wk ahead inc flu hosp	quantile	25	NA	NA	NA	0.250	NA
UMass-trends_ensemble	1 wk ahead inc flu hosp	quantile	46	NA	NA	NA	0.750	NA
UMass-trends_ensemble	1 wk ahead inc flu hosp	quantile	56	NA	NA	NA	0.900	NA
UMass-trends_ensemble	1 wk ahead inc flu hosp	quantile	68	NA	NA	NA	0.975	NA

Note: The example forecasts were made on May 15, 2023 for California at the 1 week ahead horizon. The forecasts were generated during the FluSight forecasting challenge, then formatted according to Zoltar standards for storage. The timezero, season, unit, param1, param2, and param3 columns have been omitted for brevity (The season column has a value of “2021–2022” or “2022–2023” while the last three “param” columns always have a value of NA).

TABLE 9 | An example prediction of weekly incident influenza hospitalizations.

model_id	target	horizon	output_type	output_type_id	value
UMass-trends_ensemble	wk ahead inc flu hosp	1	quantile	0.025	12
UMass-trends_ensemble	wk ahead inc flu hosp	1	quantile	0.100	17
UMass-trends_ensemble	wk ahead inc flu hosp	1	quantile	0.250	25
UMass-trends_ensemble	wk ahead inc flu hosp	1	quantile	0.750	46
UMass-trends_ensemble	wk ahead inc flu hosp	1	quantile	0.900	56
UMass-trends_ensemble	wk ahead inc flu hosp	1	quantile	0.975	68

Note: The example model output was made on May 15, 2023 for California at the 1 week ahead horizon. The forecast was generated during the FluSight forecasting challenge, then formatted according to hubverse standards post hoc. The location, forecast_date, and season columns have been omitted for brevity; quantiles representing the endpoints of the central 50%, 80% and 95% prediction intervals are shown.

above, and target week ahead incident influenza hospitalization. The values for the forecasts are always non-negative. In Table 9, we provide an example of these predictions, showing select quantiles from a single model, forecast date, horizon, and location.

Next, we can combine the predictions into a single model_out_tbl object used to generate forecasts for each ensemble method. Then, we call the appropriate function in hubEnsembles to generate predictions for each equally-weighted ensemble, storing the results in four separate objects of model output data.

```
flu_forecasts_component <- dplyr::filter(
  flu_forecasts_hubverse,
  !model_id %in% c("Flusight-baseline",
  "Flusight-ensemble"))
)

mean_ensemble <- hubEnsembles::simple_ensemble(
  flu_forecasts_component,
  weights = NULL,
  agg_fun = mean,
  model_id = "mean-ensemble"
)
median_ensemble <- hubEnsembles::simple_ensemble(
  flu_forecasts_component,
  weights = NULL,
  agg_fun = median,
  model_id = "median-ensemble"
)
lp_normal <- hubEnsembles::linear_pool(
  flu_forecasts_component,
  weights = NULL,
  n_samples = 1e5,
  model_id = "lp-normal",
  tail_dist = "norm"
)
```

```
lp_lognormal <- hubEnsembles::linear_pool(
  flu_forecasts_component,
  weights = NULL,
  n_samples = 1e5,
  model_id = "lp-lognormal",
  tail_dist = "lnorm"
)
```

We then evaluate the performance of the ensembles using scoring metrics that measure the accuracy and calibration of their forecasts. We chose several common metrics in forecast evaluation, including mean absolute error (MAE), weighted interval score (WIS) [52], 50% prediction interval (PI) coverage, and 95% PI coverage. MAE measures the average absolute error of a set of point forecasts; smaller values of MAE indicate better forecast accuracy. WIS is a generalization of MAE for probabilistic forecasts and is an alternative to other common proper scoring rules which cannot be evaluated directly for quantile forecasts [52]. WIS is made up of three component penalties: (1) for over-prediction, (2) for under-prediction, and (3) for the spread of each interval (where an interval is defined by a symmetric set of two quantiles). This metric is a weighted sum of these penalties across all prediction intervals provided. A lower WIS value indicates a more accurate forecast [52]. PI coverage provides information about whether a forecast has accurately characterized its uncertainty about future observations. The 50% PI coverage rate measures the proportion of the time that 50% prediction intervals at that nominal level included the observed value; the 95% PI coverage rate is defined similarly. Achieving approximately nominal (50% or 95%) coverage indicates a well-calibrated forecast.

We also use relative versions of WIS and MAE (rWIS and rMAE, respectively) to understand how the ensemble performance

TABLE 10 | Summary of overall model performance across both seasons, averaged over all locations except the US national location and sorted by ascending WIS.

model	wis	rwis	mae	rmae	cov50	cov95
CMU-TimeSeries	31.866	0.677	47.325	0.827	0.403	0.850
MOBS-GLEAM_FLUH	33.371	0.709	45.256	0.791	0.332	0.699
PSI-DICE	34.338	0.729	46.227	0.808	0.390	0.781
lp-normal	34.393	0.730	52.359	0.915	0.574	0.960
lp-lognormal	34.394	0.730	52.360	0.915	0.574	0.960
Median-ensemble	37.672	0.800	52.053	0.909	0.478	0.811
UMass-trends_ensemble	38.554	0.819	57.212	0.999	0.666	0.917
Mean-ensemble	39.480	0.838	54.615	0.954	0.461	0.779
SGroup-RandomForest	39.694	0.843	56.338	0.984	0.463	0.871
CU-ensemble	41.972	0.891	52.318	0.914	0.423	0.703
GT-FluFNP	42.045	0.893	51.992	0.908	0.426	0.698
SigSci-TSENS	45.865	0.974	58.231	1.017	0.485	0.726
Flusight-baseline	47.095	1.000	57.244	1.000	0.499	0.776
UVAFluX-ensemble	57.242	1.215	68.383	1.195	0.208	0.448
SigSci-CREG	60.227	1.279	67.644	1.182	0.298	0.545

Note: Both the individual FluSight component models and the resulting ensembles are included in the table, as long as they forecasted for at least 80% of the two seasons. The ensembles rank in the top half of models in terms of accuracy. The two linear pools have the best WIS and 95% coverage rates of the ensembles, as well as top performance for MAE and 50% coverage rate.

compares to that of the FluSight baseline model. These metrics are calculated as

$$rWIS = \frac{WIS_{model\ m}}{WIS_{baseline}} \quad rMAE = \frac{MAE_{model\ m}}{MAE_{baseline}},$$

where model m is any given model being compared against the baseline. For both of these metrics, a value less than one indicates better performance compared to the baseline while a value greater than one indicates worse performance. Because the baseline model is designed to represent a naive prediction, relative scores less than one indicate the model is outperforming a simple prediction that could be made in the absence of a model. By definition, the FluSight baseline itself will always have a value of one for both of these metrics.

Each unique prediction from an ensemble model is scored against target data in the oracle output format using the `score_model_out()` function from the `hubEval`s package, made for scoring hubverse model outputs with commonly used evaluation metrics including those mentioned above. We use median forecasts taken from the 0.5 quantile for the MAE evaluation. Example code scoring the ensembles' forecasts can be seen below. We attach forecasts from the baseline model in order to calculate the relative metrics.

```
flu_baseline_ensembles_scores <- flu_baseline_forecasts |>
  rbind(flu_ensembles_forecasts) |>
  hubEval::score_model_out(
    flu_oracle_output,
    metrics = c("wis", "overprediction", "underprediction",
    "dispersion", "bias",
    "ae_median", "interval_coverage_50",
    "interval_coverage_95"),
    relative_metrics = c("wis", "ae_median"),
    baseline = "Flusight-baseline",
    summarize = FALSE
  )
```

5.2 | Performance Results Across Ensembles

The four ensembles rank in the top half in terms of accuracy when compared to the individual component models that made predictions for influenza hospitalizations (Table 10). The ranking of the models differs slightly among the four scoring metrics; the two linear opinion pools perform better than the other ensembles in terms of WIS and 95% prediction interval coverage while the quantile median has slightly better MAE and the best 50% coverage across ensembles. The quantile mean displayed the worst values for every metric out of the ensembles, particularly for MAE.

Plots of the ensemble models' forecasts can aid our understanding about the origin of these accuracy differences (Figure 8). For example, the linear opinion pools consistently have some of the widest prediction intervals, and consequently the highest coverage rates. The wider intervals provided by the LOP ensembles yield the best 95% prediction interval coverage, however, this also leads to 50% interval coverage that nearly always exceeded the nominal level. Moreover, the linear pools demonstrate the strongest WIS performance at both 1st and 4th week ahead horizons during the 2022–2023 season (see Figure 9); this season was especially difficult to forecast due to its unusual timing and large peak. The wider prediction intervals prove useful in this uncertain context when other models are overconfident in their forecasts.

Although the LOP ensembles provide the best scores for WIS and 95% PI coverage, they are not always the top ensemble model. The quantile median ensemble tends to have slightly better median (point) forecasts and mid-width intervals, including the best MAE and 50% interval coverage of all of the ensembles. While the quantile median ensemble is generally well calibrated

Weekly Incident Hospitalizations for Influenza in California

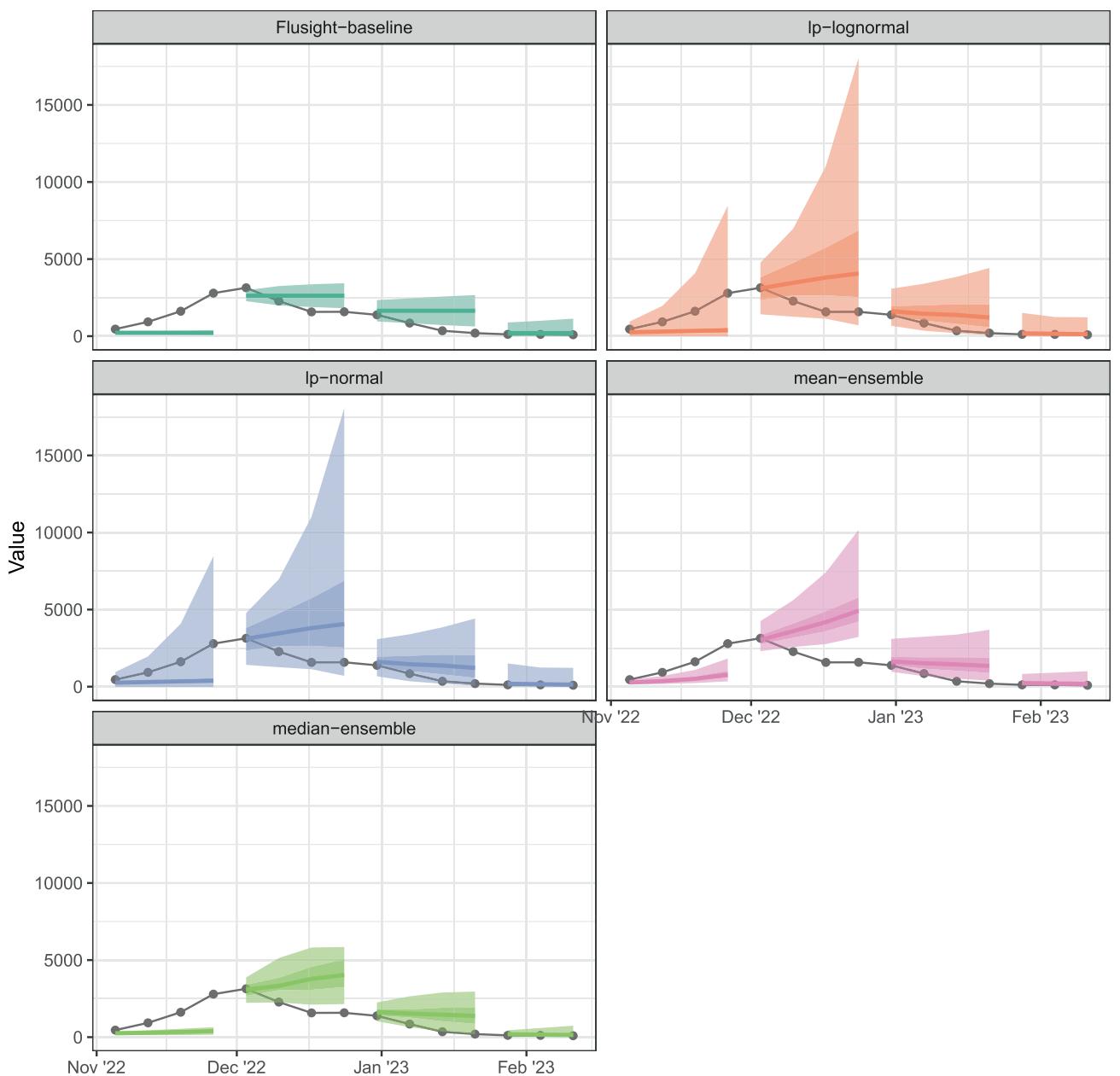


FIGURE 8 | One to four week ahead forecasts plotted against target data for California. Each panel shows the baseline and four ensembles being evaluated. All plots show the various models' forecasts summarized by the median (line) and 50% and 95% prediction intervals (darker and lighter ribbon, respectively), as compared to observed influenza hospitalizations (gray points and line). For simplicity, only predictions from select four-week periods during the 2022–2023 influenza season are shown.

for most of the predictions evaluated, it fails to appropriately capture uncertainty during the large peak of the 2022–2023 season which resulted in WIS values higher than that of the linear opinion pools (Figure 9). The quantile mean's interval widths vary throughout both seasons. Although it usually has narrower intervals than the linear pools, the quantile mean ensemble's median forecasts have a larger error margin compared to the other ensembles, especially at longer horizons. This pattern is demonstrated for the 4-week ahead forecast in California following the 2022–2023 season peak on December 5, 2022 (Figure 8). Here, the quantile mean predicted a continued increase in

hospitalizations, at a steeper slope than the other ensemble methods.

All of the ensemble variations outperform the baseline model in this analysis. The linear pool ensembles improve WIS by 27% over baseline, the quantile median by 20% and the quantile mean by 16%. Such improvements come mostly from the ensembles providing better calibrated prediction intervals, as improvements in point estimate accuracy improve over baseline by at most 9% (as measured by rMAE). While the best model (CMU-TimeSeries) improves WIS more (by 33%), ensembles are typically more

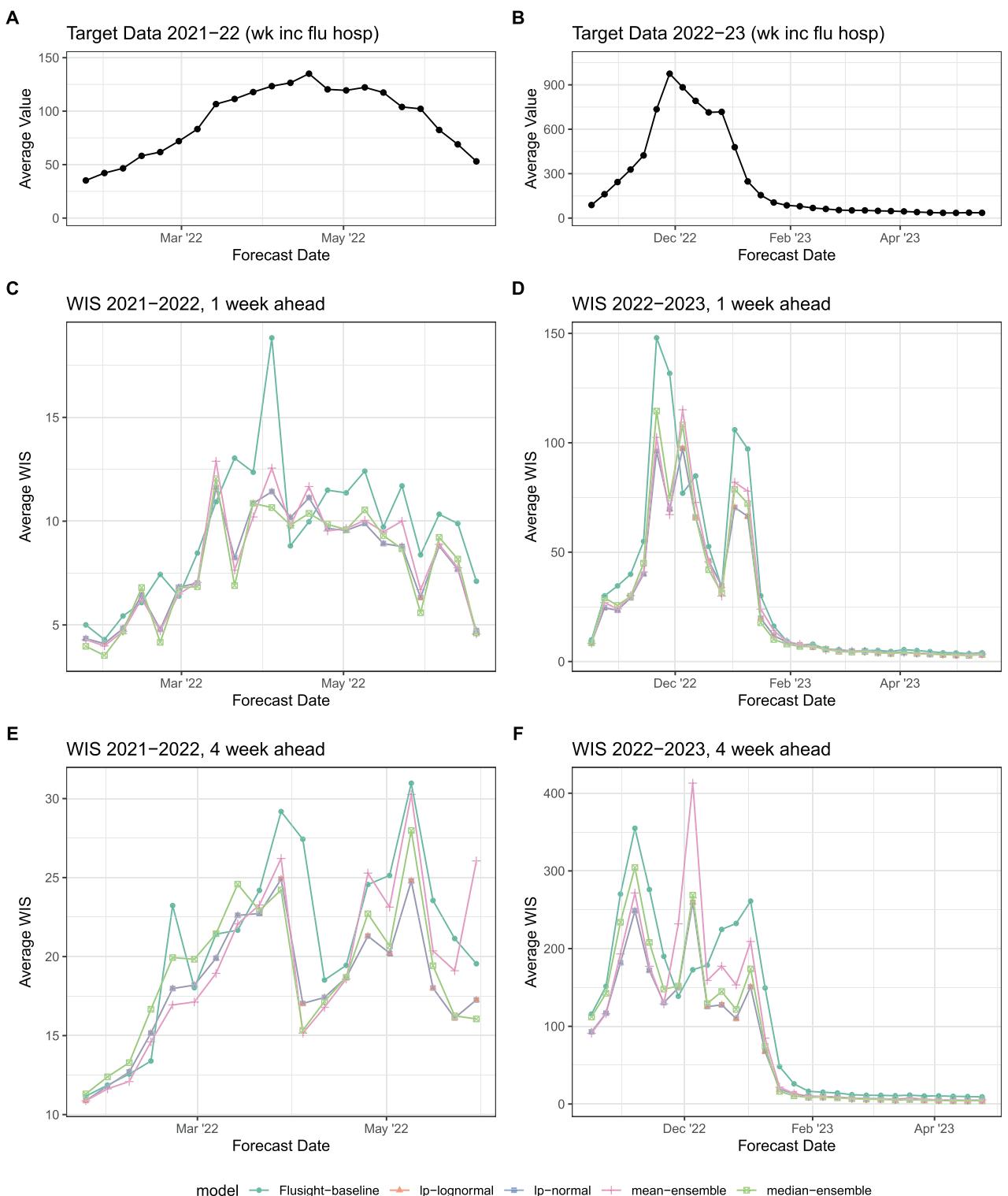


FIGURE 9 | Weighted interval score (WIS) averaged across all locations. Average target data across all locations for 2021–2022 (A) and 2022–2023 (B) seasons for reference. For each season, average WIS is shown for 1-week (C,D) and 4-week ahead (E-F) forecasts. Results are plotted for each ensemble model (colors) across the entire season. Lower values indicate better performance.

robust than individual models, and thus some decrease in overall performance may be accepted in order to avoid extreme misses.

Linear pool ensemble methods are known to generate wider, more conservative, prediction intervals than quantile averaging

methods. In this context, where influenza dynamics were perturbed by the COVID-19 pandemic interventions, the wider prediction intervals were suitable when making predictions for an influenza season with an unusually large peak. However, such wide intervals will likely be detrimental in other short-term

forecasting contexts, including, for example, during the milder season without any periods of rapid change (where the median and mean quantile ensembles demonstrated more robust performance). Expanding this evaluation to include other influenza seasons will be important to characterize the robustness of these conclusions and test how much the unusual timing and magnitude of the analyzed seasons affected forecast performance.

The choice of an appropriate ensemble aggregation method will depend on the forecast target, the goal of forecasting, and the behavior of the individual models contributing to an ensemble. One case may call for prioritizing high coverage rates while another may prioritize accurate point forecasts. The `simple_ensemble()` and `linear_pool()` functions and the ability to specify component model weights and an aggregation function for `simple_ensemble()` allow users to implement a variety of ensemble methods.

6 | Summary and Discussion

Ensembles of independent models are a powerful tool to generate more accurate and more reliable predictions of future outcomes than those from a single model alone. Here, we have provided an overview of multi-model ensemble methodology with the goal of improving use of ensemble results in public health and biomedical research settings, as well as other domains that use probabilistic forecasting. Moreover, we have demonstrated how to utilize `hubEnsembles`, a simple and flexible framework to combine individual model predictions into an ensemble.

Multi-model ensembles are becoming the gold standard for prediction exercises in the public health domain. Collaborative modeling hubs can serve as a centralized entity to guide and elicit predictions from multiple independent models, as well as to generate and communicate ensemble results [12, 36]. Given the increasing popularity of multi-model ensembles and collaborative hubs, there is a clear need for generalized data standards and software infrastructure to support these hubs. By addressing this need, the hubverse suite of tools can reduce duplicative efforts across existing hubs, support other communities engaged in collaborative efforts, and enable the adoption of multi-model approaches in new domains.

When generating and interpreting an ensemble prediction, it is important to understand the methods underlying the ensemble, as methodological choices can have meaningful effects on the resulting ensemble and its performance. Although there may not be a universal “best” method, matching the properties of a given ensemble method with the features of the component models will likely yield best results [24]. Our case study on seasonal influenza forecasts in the US demonstrates this point. The two linear opinion pools with different distributional tails performed best for many of the metrics we evaluated—weighted interval score, mean absolute error, and prediction interval coverage—particularly when outlying component forecasts were important. Yet, the quantile averaging methods outperformed the linear pools during periods of greater stability and less variation. All ensembles showed clear improvement over the baseline model, demonstrating the accuracy gains from ensemble models that motivate the use of a “hub-based” approach to prediction for infectious diseases, public health, and in other fields.

Acknowledgments

The authors thank all members of the hubverse community; the broader hubverse software infrastructure made this package possible. Li Shandross, Anna Krystalli, Nicholas G. Reich, and Evan L. Ray were supported by the National Institutes of General Medical Sciences (NIGMS, R35GM119582) and the US Centers for Disease Control and Prevention (U01IP001122 and NU38FT000008). Emily Howerton was supported by NSF RAPID awards DEB-2126278 and DEB-2220903, as well as the Eberly College of Science Barbara McClintock Science Achievement Graduate Scholarship in Biology at the Pennsylvania State University. Lucie Contamin and Harry Hochheiser were supported by NIGMS grants U24GM132013 and R24GM153920. The content is solely the responsibility of the authors and does not necessarily represent the official views of NIGMS, the National Institutes of Health, or CDC.

Consortium of Infectious Disease Modeling Hubs authors include Alvaro J. Castro Rivadeneira (University of Massachusetts Amherst), Lucie Contamin (University of Pittsburgh), Sebastian Funk (London School of Hygiene & Tropical Medicine), Aaron Gerding (University of Massachusetts Amherst), Hugo Gruson (data.org), Harry Hochheiser (University of Pittsburgh), Emily Howerton (The Pennsylvania State University), Melissa Kerr (University of Massachusetts Amherst), Anna Krystalli (R-RSE SMPC), Sara L. Loo (Johns Hopkins University), Evan L. Ray (University of Massachusetts Amherst), Nicholas G. Reich (University of Massachusetts Amherst), Koji Sato (Johns Hopkins University), Li Shandross (University of Massachusetts Amherst), Katharine Sherratt (London School of Hygiene and Tropical Medicine), Shaun Truelove (Johns Hopkins University), Martha Zorn (University of Massachusetts Amherst).

Conflicts of Interest

N.G.R. reports consulting income from Google Research.

Data Availability Statement

The data that support the findings of this study are available in `hubEnsembles` Manuscript at <https://github.com/hubverse-org/hubEnsemblesManuscript/>. These data were derived from the following resources available in the public domain: CDC FluSight Forecasting Challenge, <https://zoltardata.com/project/299>. All code necessary to reproduce this manuscript can be found in a GitHub repository available at <https://github.com/hubverse-org/hubEnsemblesManuscript> and archived at <https://doi.org/10.5281/zenodo.17259775>, including required package versions which are available in the `renv.lock` file.

Endnotes

¹ https://hubverse-org.github.io/hubUtils/reference/as_model_out_tbl.html.

References

1. R. T. Clemen, “Combining Forecasts: A Review and Annotated Bibliography,” *International Journal of Forecasting* 5, no. 4 (1989): 559–583, [https://doi.org/10.1016/0169-2070\(89\)90012-5](https://doi.org/10.1016/0169-2070(89)90012-5).
2. A. Timmermann, “Chapter 4 Forecast Combinations,” in *Handbook of Economic Forecasting*, vol. 1 (Elsevier, 2006), 135–196, [https://doi.org/10.1016/S1574-0706\(05\)01004-9](https://doi.org/10.1016/S1574-0706(05)01004-9).
3. M. Hibon and T. Evgeniou, “To Combine or Not to Combine: Selecting Among Forecasts and Their Combinations,” *International Journal of Forecasting* 21, no. 1 (2005): 15–24, <https://doi.org/10.1016/j.ijforecast.2004.05.002>.
4. R. B. Alley, K. A. Emanuel, and F. Zhang, “Advances in Weather Prediction,” *Science* 363, no. 6425 (2019): 342–344, <https://doi.org/10.1126/science.aav7274>.

5. C. Tebaldi and R. Knutti, "The Use of the Multi-Model Ensemble in Probabilistic Climate Projections," *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences* 365, no. 1857 (2007): 2053–2075, <https://doi.org/10.1098/rsta.2007.2076>.

6. K. A. Aastveit, J. Mitchell, F. Ravazzolo, and H. K. van Dijk, *The Evolution of Forecast Density Combinations in Economics, Tinbergen Institute Discussion Paper, No. TI 2018-069/III* (Tinbergen Institute, 2018).

7. J. Shaman and S. Kandula, "Improved Discrimination of Influenza Forecast Accuracy Using Consecutive Predictions," *PLOS Currents* (2015), <https://doi.org/10.1371/currents.outbreaks.8a6a3df285af7ca973fab4b22e10911e>.

8. L. Held, S. Meyer, and J. Bracher, "Probabilistic Forecasting in Infectious Disease Epidemiology: The 13th Armitage Lecture," *Statistics in Medicine* 36, no. 22 (2017): 3443–3460, <https://doi.org/10.1002/sim.7363>.

9. E. L. Ray, K. Sakrejda, S. A. Lauer, M. A. Johansson, and N. G. Reich, "Infectious Disease Prediction With Kernel Conditional Density Estimation," *Statistics in Medicine* 36, no. 30 (2017): 4908–4929, <https://doi.org/10.1002/sim.7488>.

10. N. G. Reich, C. J. McGowan, T. K. Yamana, et al., "Accuracy of Real-Time Multi-Model Ensemble Forecasts for Seasonal Influenza in the U.S.," *PLoS Computational Biology* 15, no. 11 (2019): e1007486, <https://doi.org/10.1371/journal.pcbi.1007486>.

11. A. Rodríguez, H. Kamarthi, P. Agarwal, et al., "Machine Learning for Data-Centric Epidemic Forecasting," *Nature Machine Intelligence* 6 (2024): 1122–1131, <https://doi.org/10.1038/s42256-024-00895-7>.

12. N. G. Reich, J. Lessler, S. Funk, et al., "Collaborative Hubs: Making the Most of Predictive Epidemic Modeling," *American Journal of Public Health* 112, no. 6 (2022): 839–842, <https://doi.org/10.2105/AJPH.2022.306831>.

13. C. J. McGowan, M. Biggerstaff, M. Johansson, et al., "Collaborative Efforts to Forecast Seasonal Influenza in the United States, 2015–2016," *Scientific Reports* 9, no. 1 (2019): 683, <https://doi.org/10.1038/s41598-018-36361-9>.

14. M. A. Johansson, K. M. Apfeldorf, S. Dobson, et al., "An Open Challenge to Advance Probabilistic Forecasting for Dengue Epidemics," *Proceedings of the National Academy of Sciences of the United States of America* 116, no. 48 (2019): 24268–24274, <https://doi.org/10.1073/pnas.1909865116>.

15. K. M. Holcomb, S. Mathis, J. E. Staples, et al., "Evaluation of an Open Forecasting Challenge to Assess Skill of West Nile Virus Neuroinvasive Disease Prediction," *Parasites & Vectors* 16, no. 1 (2023): 11, <https://doi.org/10.1186/s13071-022-05630-y>.

16. E. Y. Cramer, E. L. Ray, V. K. Lopez, et al., "Evaluation of Individual and Ensemble Probabilistic Forecasts of COVID-19 Mortality in the United States," *Proceedings of the National Academy of Sciences of the United States of America* 119, no. 15 (2022): e2113561119, <https://doi.org/10.1073/pnas.2113561119>.

17. K. Sherratt, H. Gruson, R. Grah, et al., "Predictive Performance of Multi-Model Ensemble Forecasts of COVID-19 Across European Nations," *eLife* 12 (2023): e81916, <https://doi.org/10.7554/eLife.81916>.

18. E. Howerton, L. Contamin, L. C. Mullany, et al., "Evaluation of the US COVID-19 Scenario Modeling Hub for Informing Pandemic Response Under Uncertainty," *Nature Communications* 14, no. 1 (2023): 7260, <https://doi.org/10.1038/s41467-023-42680-x>.

19. T. K. Yamana, S. Kandula, and J. Shaman, "Superensemble Forecasts of Dengue Outbreaks," *Journal of the Royal Society Interface* 13, no. 123 (2016): 20160410, <https://doi.org/10.1098/rsif.20160410>.

20. E. L. Ray and N. G. Reich, "Prediction of Infectious Disease Epidemics via Weighted Density Ensembles," *PLoS Computational Biology* 14, no. 2 (2018): e1005910, <https://doi.org/10.1371/journal.pcbi.1005910>.

21. C. Viboud, K. Sun, R. Gaffey, et al., "The RAPIDD Ebola Forecasting Challenge: Synthesis and Lessons Learnt," *Epidemics* 22 (2018): 13–21, <https://doi.org/10.1016/j.epidem.2017.08.002>.

22. F. J. Colón-González, L. S. Bastos, B. Hofmann, et al., "Probabilistic Seasonal Dengue Forecasting in Vietnam: A Modelling Study Using Superensembles," *PLoS Medicine* 18, no. 3 (2021): e1003542, <https://doi.org/10.1371/journal.pmed.1003542>.

23. J. Paireau, A. Andronico, N. Hozé, et al., "An Ensemble Model Based on Early Predictors to Forecast COVID-19 Health Care Demand in France," *Proceedings of the National Academy of Sciences of the United States of America* 119, no. 18 (2022): e2103302119, <https://doi.org/10.1073/pnas.2103302119>.

24. E. Howerton, M. C. Runge, T. L. Bogich, et al., "Context-Dependent Representation of Within- and Between-Model Uncertainty: Aggregating Probabilistic Predictions in Infectious Disease Epidemiology," *Journal of the Royal Society Interface* 20, no. 198 (2023): 20220659, <https://doi.org/10.1098/rsif.2022.0659>.

25. E. L. Ray, L. C. Brooks, J. Bien, et al., "Comparing Trained and Untrained Probabilistic Ensemble Forecasts of COVID-19 Cases and Deaths in the United States," *International Journal of Forecasting* 39, no. 3 (2023): 1366–1383, <https://doi.org/10.1016/j.ijforecast.2022.06.005>.

26. F. Pedregosa, G. Varoquaux, A. Gramfort, et al., "Scikit-Learn: Machine Learning in Python," *Journal of Machine Learning Research* 12, no. 85 (2011): 2825–2830, <https://doi.org/10.5555/1953048.2078195>.

27. C. E. Weiss, E. Raviv, and G. Roetzer, "Forecast Combinations in R Using the ForecastComb Package," *R Journal* 10, no. 2 (2019): 262, <https://doi.org/10.32614/RJ-2018-052>.

28. N. Bosse, Y. Yao, S. Abbott, and S. Funk, "Stackr Create Mixture Models From Predictive Samples," 2023.

29. S. Couch and M. Kuhn, "Stack Tidy Model Stacking," 2023.

30. Consortium of Infectious Disease Modeling Hubs, "The hubverse: Open tools for collaborative forecasting," 2025, <https://docs.hubverse.io/en/latest/index.html>.

31. S. B. Vincent, "The Function of the Vibrissae in the Behavior of the White Rat. PhD thesis. University of Chicago," 1912.

32. M. Stone, "The Opinion Pool," *Annals of Mathematical Statistics* 32, no. 4 (1961): 1339–1342.

33. K. C. Lichtendahl, Y. Grushka-Cockayne, and R. L. Winkler, "Is It Better to Average Probabilities or Quantiles?," *Management Science* 59, no. 7 (2013): 1594–1611, <https://doi.org/10.1287/mnsc.1120.1667>.

34. T. D. Hollingsworth and G. F. Medley, "Learning From Multi-Model Comparisons: Collaboration Leads to Insights, but Limitations Remain," *Epidemics* 18 (2017): 1–3, <https://doi.org/10.1016/j.epidem.2017.02.014>.

35. K. Shea, M. C. Runge, D. Pannell, et al., "Harnessing Multiple Models for Outbreak Management," *Science* 368, no. 6491 (2020): 577–579, <https://doi.org/10.1126/science.abb9934>.

36. R. K. Borchering, J. M. Healy, B. L. Cadwell, et al., "Public Health Impact of the U.S. Scenario Modeling Hub," *Epidemics* 44 (2023): 100705, <https://doi.org/10.1016/j.epidem.2023.100705>.

37. N. G. Reich, L. C. Brooks, S. J. Fox, et al., "A Collaborative Multiyear, Multimodel Assessment of Seasonal Influenza Forecasting in the United States," *Proceedings of the National Academy of Sciences of the United States of America* 116, no. 8 (2019): 3146–3154, <https://doi.org/10.1073/pnas.2200703119>.

38. R. K. Borchering, L. C. Mullany, E. Howerton, et al., "Impact of SARS-CoV-2 Vaccination of Children Ages 5–11 Years on COVID-19 Disease Burden and Resilience to New Variants in the United States, November 2021–March 2022: A Multi-Model Study," *Lancet Regional Health – Americas* 17 (2023): 100398, <https://doi.org/10.1016/j.lana.2022.100398>.

39. S. L. Loo, E. Howerton, L. Contamin, et al., "The US COVID-19 and Influenza Scenario Modeling Hubs: Delivering Long-Term Projections to Guide Policy," *Epidemics* 46 (2024): 100738, <https://doi.org/10.1016/j.epidem.2023.100738>.

40. S. Jung, S. L. Loo, E. Howerton, et al., "Potential Impact of Annual Vaccination With Reformulated COVID-19 Vaccines: Lessons From the US COVID-19 Scenario Modeling Hub," *PLoS Medicine* 21, no. 4 (2024): e1004387, <https://doi.org/10.1371/journal.pmed.1004387>.

41. N. E. Dean, A. Pastore y Piontti, Z. J. Madewell, et al., "Ensemble Forecast Modeling for the Design of COVID-19 Vaccine Efficacy Trials," *Vaccine* 38, no. 46 (2020): 7213–7216, <https://doi.org/10.1016/j.vaccine.2020.09.031>.

42. S. Pollett, M. A. Johansson, N. G. Reich, et al., "Recommended Reporting Items for Epidemic Forecasting and Prediction Research: The EPI-FORGE 2020 Guidelines," *PLoS Medicine* 18, no. 10 (2021): e1003793, <https://doi.org/10.1371/journal.pmed.1003793>.

43. R. J. Oidtmann, E. Omodei, M. U. G. Kraemer, et al., "Trade-Offs Between Individual and Ensemble Forecasts of an Emerging Infectious Disease," *Nature Communications* 12, no. 1 (2021): 5379, <https://doi.org/10.1038/s41467-021-25695-0>.

44. N. Wattanachit, E. L. Ray, T. C. McAndrew, and N. G. Reich, "Comparison of Combination Methods to Create Calibrated Ensemble Forecasts for Seasonal Influenza in the U.S.," *Statistics in Medicine* 42, no. 26 (2023): 4696–4712, <https://doi.org/10.1002/sim.9884>.

45. T. McAndrew and N. G. Reich, "Adaptively Stacking Ensembles for Influenza Forecasting," *Statistics in Medicine* 40, no. 30 (2021): 6931–6952, <https://doi.org/10.1002/sim.9219>.

46. S. J. Fox, M. Kim, L. A. Meyers, N. G. Reich, and E. L. Ray, "Optimizing Disease Outbreak Forecast Ensembles," *Emerging Infectious Diseases* 30, no. 9 (2024): 1967–1969, <https://doi.org/10.3201/eid3009.240026>.

47. K. Sherratt, A. Srivastava, K. Ainslie, et al., "Characterising Information Gains and Losses When Collecting Multiple Epidemic Model Outputs," *Epidemics* 47 (2024): 100765, <https://doi.org/10.1016/j.epidem.2024.100765>.

48. E. L. Ray and A. Gerdin, "Distfromq: Reconstruct a Distribution from a Collection of Quantiles," 2024.

49. H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods* (Society for Industrial and Applied Mathematics, 1992).

50. CDC, "About Flu Forecasting," 2023, <https://www.cdc.gov/flu/weekly/flusight/how-flu-forecasting.htm>.

51. N. G. Reich, M. Cornell, E. L. Ray, K. House, and K. Le, "The Zoltar Forecast Archive, a Tool to Standardize and Store Interdisciplinary Prediction Research," *Scientific Data* 8, no. 1 (2021): 59, <https://doi.org/10.1038/s41597-021-00839-5>.

52. J. Bracher, E. L. Ray, T. Gneiting, and N. G. Reich, "Evaluating Epidemic Forecasts in an Interval Format," *PLoS Computational Biology* 17, no. 2 (2021): e1008618, <https://doi.org/10.1371/journal.pcbi.1008618>.