Multi-model ensembles in infectious disease and public health: methods, interpretation, and implementation in R

Li Shandross   Emily Howerton   Lucie Contamin
Harry Hochheiser   Anna Krystalli
Consortium of Infectious Disease Modeling Hubs
Nicholas G. Reich   Evan L. Ray

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 diveristy 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` pacakge, 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 US COVID-19 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 utility for public health officials and medical practitioners. Underlying these predictions are mathematical models that use historical disease incidence data to make probabilistic predictions of incidence in the future[7–9] [**EH: suggestions on references here would be great**]. Given the performance benefits of multi-model ensembles, it is becoming increasingly common to convene multiple modeling teams into a collaborative "hub"[10], 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[11,12], dengue[13], West Nile virus[14], and more recently SARS-CoV-2[15–17].

Generating multi-model ensmebles or interpreting results 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[18–24], yet there is no consensus on which method should be favored. There are software packages that support various aspects of multi-model ensembling[25–28]. 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. The `hubEnsembles` package provides a flexible framework for generating ensemble predictions from multiple models across a range of common methods and prediction types; 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[29]. These two factors together, a simple implementation framework across methods and integration with hubverse data standards and tools, makes `hubEnsembles` accessible and easy to use. Finally, we present a basic demonstration of multi-model ensemble generation and interpretion (Section 4), and a more in-depth analysis using real influenza forecasts (Section 5). Together, these case studies motivate a discussion and comparison of the various methods (Section 6). 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 for public health applications.

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 basic definitions of key concepts in statistical forecasting, and an overview of the classes of methods that are typically used for generating a multi-model ensemble.

2.1 Key concepts in statistical 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 being combined 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 (in this case, the generation interval for an infectious pathogen). Throughout, we will use the general term "prediction" to encapsulate all such outcomes that could be combined. 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 use a weight w_i for each prediction; we define the set of model-specific weights as $\boldsymbol{w} = \{w_i | i \in 1, ..., N\}$. 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, $\boldsymbol{p} = \{p_i | i \in 1, ..., N\}$, each from a distinct model i, an ensemble of these predictions is

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

using any function C and any set of model-specific weights $\boldsymbol{w}$. 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
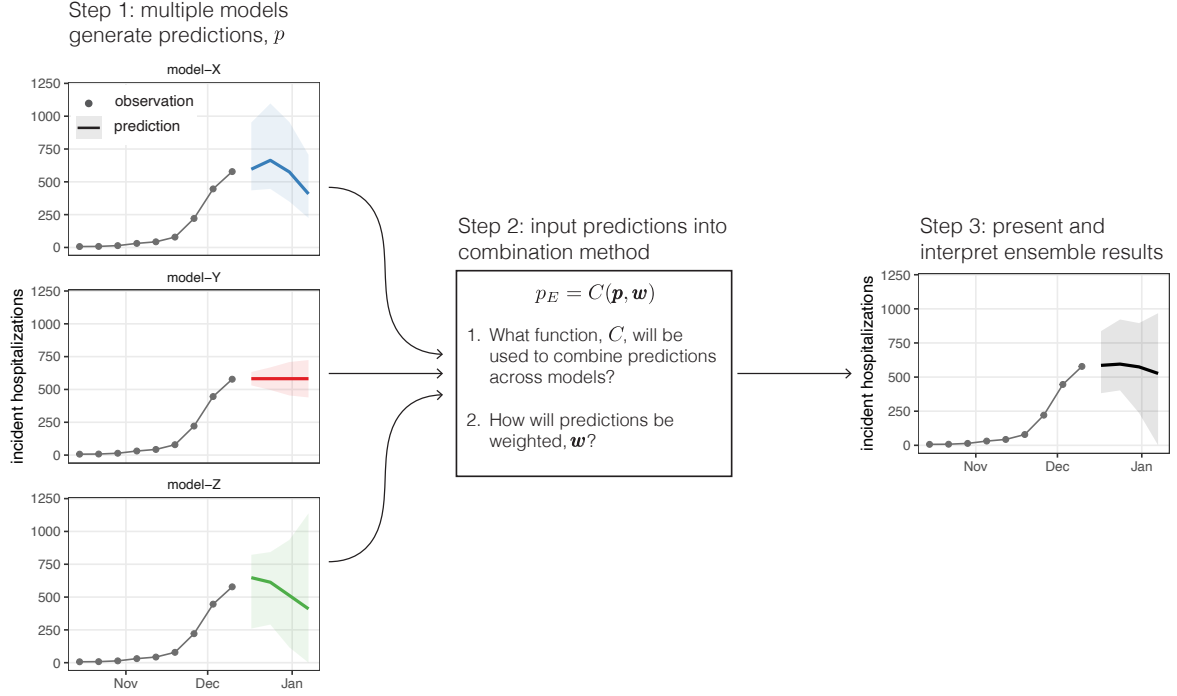
Figure 1: Overview of process to generate a multi-model ensemble. Predictions, p_i are generated from N independent models (step 1). Then those predictions, $\boldsymbol{p} = \{p_i | i \in 1, ..., N\}$, are combined with some function, C, and set of weights, $\boldsymbol{w} = \{w_i | i \in 1, ..., N\}$. This figure illustrates example probabilistic forecasts for incidence influenza hospitalizations, where the median (line) and 90/% prediction interval are shown. The resulting ensemble uses the linear pool method ($F_{LOP}(x)\}$).

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[30]) and probability averaging (also called a distributional mixture or linear opinion pool[31])[32]. 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, $\mathcal{Q} = \{F_i^{-1}(\theta) | i \in 1, ..., N\}$, with a given set of weights, $\boldsymbol{w}$, as

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

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 shape and scale parameter, the resulting quantile average will be the same type of distribution, with shape and scale parameters that are the average of the shape 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 pool is calculated by averaging probabilities across predictions for a fixed value of the target variable, x. In other words, for a set $\mathcal{F} = \{F_i(x) | i \in 1, ..., N\}$ containing the values of CDFs at the point x and weights $\boldsymbol{w}$, the linear pool is calculated as

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

For a set of PMF values, $\{f_i(x) | i \in 1, ..., N\}$, the linear pool can be equivalently calculated: $f_{LOP}(x) = \sum_{i=1}^{N} w_i f_i(x)$. 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. 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 $\sigma^2_{LOP} = \sum_{i=1}^{N} w_i \sigma_i^2 + \sum_{i=1}^{N} w_i (\mu_i - \mu_{LOP})^2$, 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[32]. 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[32]. The linear pool method preserves variation between individual models, whereas the quantile average cancels away this variation under the assumption it constitutes sampling error[23].

2.3 Applications in public health and infectious disease outbreaks

[**EH: THOUGHTS/EDITS TO THIS SECTION ARE WELCOME!**]

Multi-model ensembles have become the gold standard for forecasting and prediction efforts that support public health in real time[10,33–35]. One prominent domain is forecasting key characteristics of infectious disease outbreaks, including weekly incidence or healthcare demand over future weeks, disease burden for the entire season, and timing and magnitude of the outbreak peak[13–15,22,36]. Projections of disease outcomes under multiple possible future scenarios have also been used to estimate intervention effectiveness to inform policy[37–39], and it has been proposed to use short-term forecasts of incidence to inform vaccine efficacy trials[40]. Standard guidelines for reporting of prediction efforts in outbreak and public health settings have also been established[41].

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[11–13,15–18,22]. Notably, the ensemble does not always outperform the best model, but typically offers improved consistency and robustness over individual models[15,16,19,42]. However, in one instance, a baseline historical average of West Nile Virus cases in the US outperformed most model predictions, including the ensemble[14].

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[43] and dynamic weighting[44] 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 individuals models[24], and was thus used as the primary ensemble for short-term forecasts of COVID-19[15,16]. For longer-term predictions of COVID-19, a trimmed LOP ensemble performed best, as models tended to be more overconfident in this setting[17]. The number of models submitting real-time predictions has varied dramatically (from as few as four models for longer-term predictions of COVID-19[17] to more than 40 for short-term forecasts of COVID-19[15]), research across influenza and

COVID-19 suggests that at least 3 models are needed, with diminishing returns for every model that is added[45].

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.

3 How to implement ensemble calculations

The methods described in Section 2 are implemented in 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. 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, then explain the implementation of the two primary ensembling functions included in the package, `simple_ensemble()` and `linear_pool()`.

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 fixed, the task ID variables may vary according to the needs of the modeling hub or modeling task[29].

Table 1 provides an example of model output that stores short-term forecasts of weekly flu 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).

Table 1: Example of forecasts for incident influenza hospitalizations, formatted according to hubverse standards. Quantile predictions for the median and 50% prediction intervals from a single model are shown for four distct 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_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

As mentioned previously, task ID variables are not fixed in name, number, or composition to incorporate flexibility in the `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[29].

The "model output representation" columns `output_type` and `output_type_id` contain metadata about how the predictions are conveyed and are always the same. 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.

Table 2: A table summarizing how the model output representation columns are used for predictions of different output types. (*The sample output type is unique in that it can be used to capture dependence in the modeled outcomes across levels of task ID variables.) Adapted from https://hubverse.io/en/latest/user-guide/model-output.html#formats-of-model-output

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 discrete outcome variable
sample*	Integer or string specifying the sample index	Numeric: A sample from the predictive distribution

All output types can summarize predictions from marginal distributions. The sample output type is unique in that it can additionally represent predictions from joint predictive distributions. This means that the sample representation may encode dependence across combinations of multiple values for modeled task ID variables when recording samples from a joint predictive distribution. In this case, 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 "truth" data, which can be stored in one of two target data 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. Oracle output, on the other hand, represents prediction that and "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 post hoc. 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 3: Summary of ensemble function calculations for each output type. 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.

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

output_type	simple_ensemble(..., agg_fun=mean)	linear_pool()
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

3.2.1 Simple ensemble

The `simple_ensemble()` function directly computes an ensemble from component model outputs by combining them via some 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. 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. Currently, this function can be used to combine predictions with output types mean, quantile, CDF, PMF, and sample. Unlike `simple_ensemble()`, this function handles its computation 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).

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.

For the quantile output type, implementation of LOP is comparatively less straightforward. 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[46] 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[47].
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
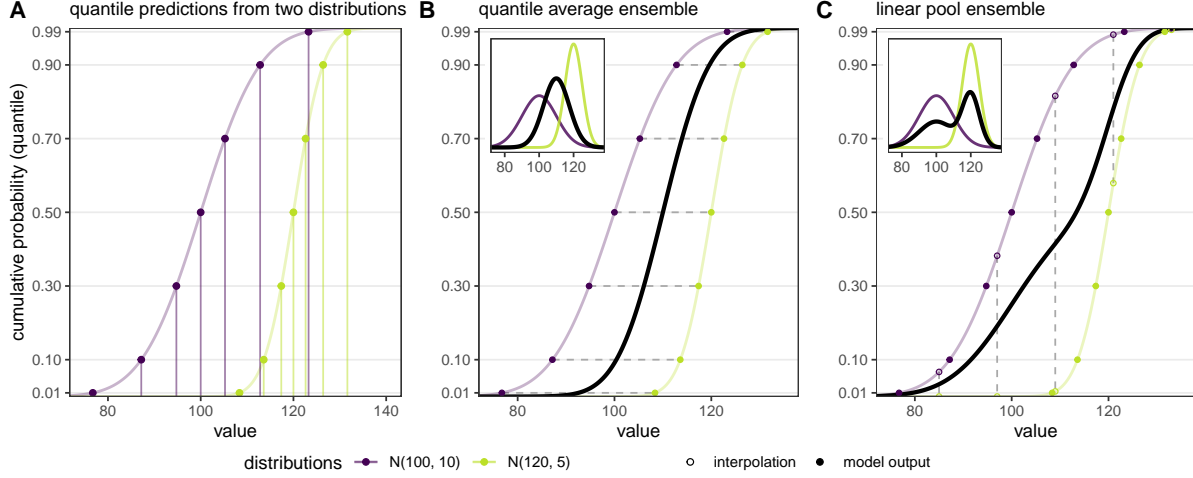
Figure 2: (Panel A) Example of quantile output type predictions. Solid points show model output collected for seven fixed quantile levels ($\theta = 0.01$, 0.1, 0.3, 0.5, 0.7, 0.9, and 0.99) from two distributions ($N(100, 10)$ in purple and $N(120, 5)$ in green), with the underlying cumulative distribution functions (CDFs) shown with curves. The y-axis ticks show each of the fixed quantile levels. The associated values for each fixed quantile level do not align across distributions (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, the model outputs are used to interpolate the full CDF for each distribution from which quantiles can be extracted for fixed values (shown with open circles). The black line shows the resulting linear pool average ensemble. Inset shows corresponding PDFs.

13

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`.

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 observational 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 made for 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 (a combination of the date the forecast was made and the forecast horizon, `target_end_date`), 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.

Table 4: Example model output for forecasts of weekly incident influenza hospitalizations. 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, and 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.

model_id	target	horizon	output_type	output_type_id	value
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	median	NA	582.00
Flusight-baseline	wk inc flu hosp	1	mean	NA	582.07

Table 4: Example model output for forecasts of weekly incident influenza hospitalizations. 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, and 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.

model_id	target	horizon	output_type	output_type_id	value
MOBS-GLEAM_FLUH	wk inc flu hosp	1	quantile	0.05	446.00
MOBS-GLEAM_FLUH	wk inc flu hosp	1	quantile	0.25	563.00
MOBS-GLEAM_FLUH	wk inc flu hosp	1	quantile	0.75	803.00
MOBS-GLEAM_FLUH	wk inc flu hosp	1	quantile	0.95	1097.00
MOBS-GLEAM_FLUH	wk inc flu hosp	1	median	NA	664.00
MOBS-GLEAM_FLUH	wk inc flu hosp	1	mean	NA	704.73
PSI-DICE	wk inc flu hosp	1	quantile	0.05	290.00
PSI-DICE	wk inc flu hosp	1	quantile	0.25	496.00
PSI-DICE	wk inc flu hosp	1	quantile	0.75	712.00
PSI-DICE	wk inc flu hosp	1	quantile	0.95	843.00
PSI-DICE	wk inc flu hosp	1	median	NA	613.00
PSI-DICE	wk inc flu hosp	1	mean	NA	594.46

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 incident influenza hospitalizations in a given week for given location using columns `observation`, `date`, and `location`. 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.

Table 5: Example target time series data for incident influenza hospitalizations. This table includes target time series data from 2022-11-01 and 2023-02-01. The target data is provided in the `hubExamples` package.

date	location	observation
2022-11-05	25	31
2022-11-12	25	43

Table 5: Example target time series data for incident influenza hospitalizations. This table includes target time series data from 2022-11-01 and 2023-02-01. The target data is provided in the `hubExamples` package.

date	location	observation
2022-11-19	25	79
2022-11-26	25	221
2022-12-03	25	446
2022-12-10	25	578
2022-12-17	25	694
2022-12-24	25	769
2022-12-31	25	733
2023-01-07	25	466
2023-01-14	25	238
2023-01-21	25	122
2023-01-28	25	71

We can plot these forecasts and the target time series data (Figure 3) 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 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(
+     location == "25",
+     date >= "2022-11-01", date <= "2023-02-01"
+   )
>
> hubVis::plot_step_ahead_model_output(
+   model_out_tbl = model_outputs_plot,
+   target_data = target_data_plot,
+   facet = "model_id",
+   facet_nrow = 1,
```

16

```
+    interactive = FALSE,
+    intervals = c(0.5, 0.9),
+    show_legend = FALSE,
+    use_median_as_point = TRUE,
+    x_col_name = "target_end_date",
+    x_target_col_name = "date"
+ ) +
+    theme_bw() +
+    labs(y = "incident hospitalizations")
```
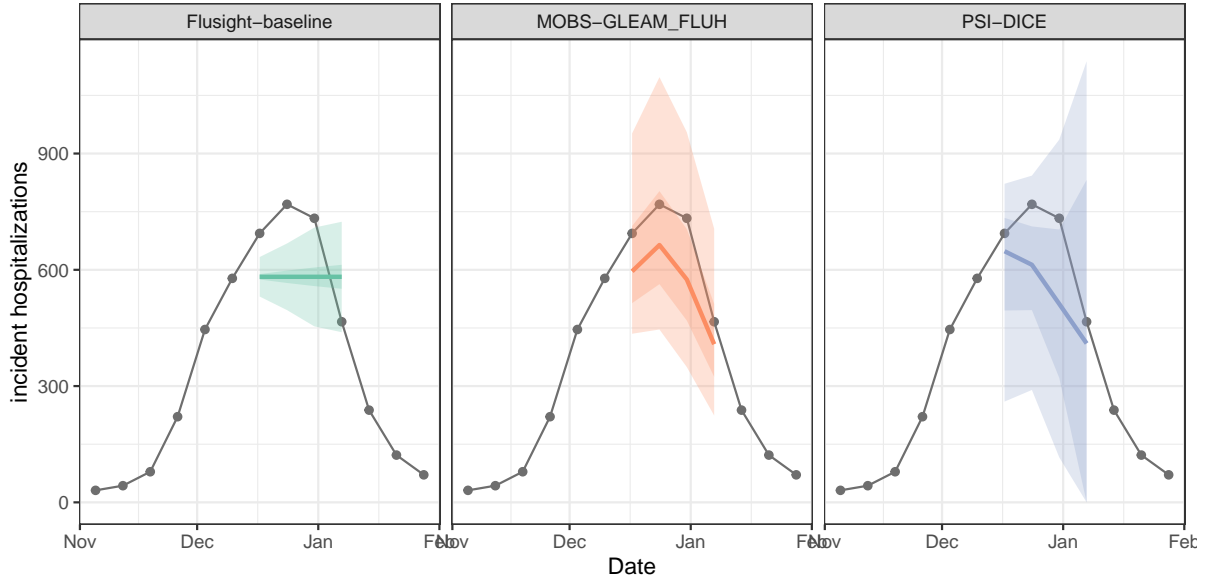

Figure 3: One example set of quantile forecasts for weekly incident influenza hospitalizations in Massachusetts from each of three models (panels). Forecasts are represented by a median (line), 50% and 90% prediction intervals (ribbons). Gray points represent observed incident hospitalizations.

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.

Table 6: Example PMF model output for forecasts of incident influenza hospitalization intensity. 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.

model_id	target	horizon	output_type	output_type_id	value
Flusight-baseline	wk flu hosp rate category	1	pmf	low	0.00
Flusight-baseline	wk flu hosp rate category	1	pmf	moderate	0.00
Flusight-baseline	wk flu hosp rate category	1	pmf	high	0.07
Flusight-baseline	wk flu hosp rate category	1	pmf	very high	0.92
MOBS-GLEAM_FLUH	wk flu hosp rate category	1	pmf	low	0.00
MOBS-GLEAM_FLUH	wk flu hosp rate category	1	pmf	moderate	0.00
MOBS-GLEAM_FLUH	wk flu hosp rate category	1	pmf	high	0.16
MOBS-GLEAM_FLUH	wk flu hosp rate category	1	pmf	very high	0.83
PSI-DICE	wk flu hosp rate category	1	pmf	low	0.01
PSI-DICE	wk flu hosp rate category	1	pmf	moderate	0.07
PSI-DICE	wk flu hosp rate category	1	pmf	high	0.22
PSI-DICE	wk flu hosp rate category	1	pmf	very high	0.70

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 4).
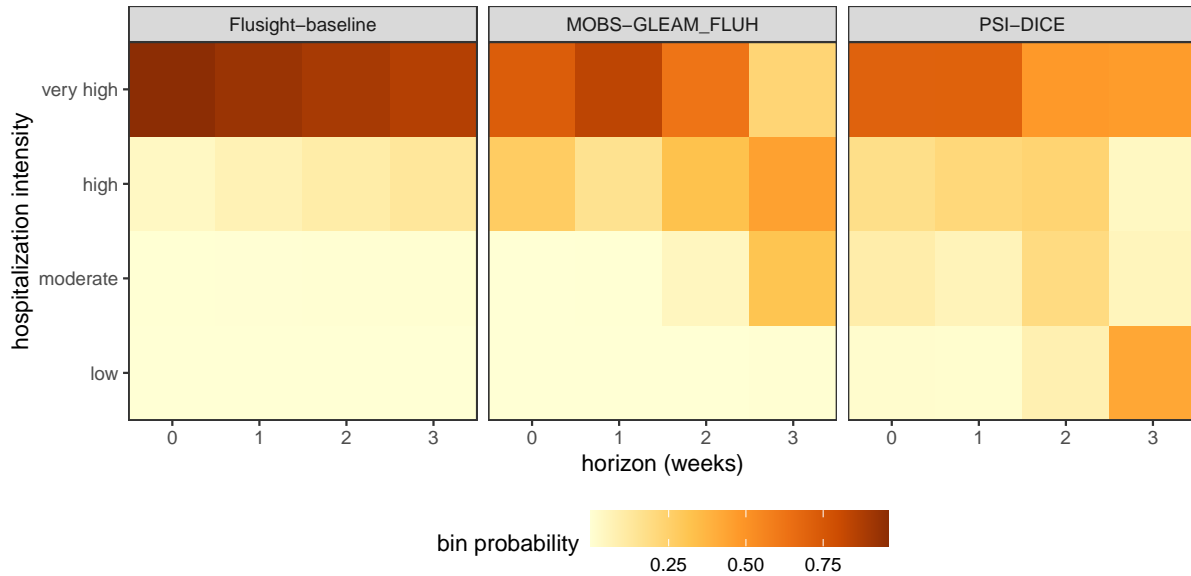
Figure 4: One example PMF forecast of incident influenza hospitalization intensity is shown for each of three models (panels). Each cell shows the forecasted probability of a given hospitalization intensity bin (low, moderate, high, and very high) for each forecast horizon (0-3 weeks ahead). Darker colors indicate higher forecasted probability.

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 uses different 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 mean, median, CDF and PMF output types, the ensemble is a linear pool.

```
> 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".

19

Table 7: Mean ensemble model output. 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 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.

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

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 5 demonstrate this.
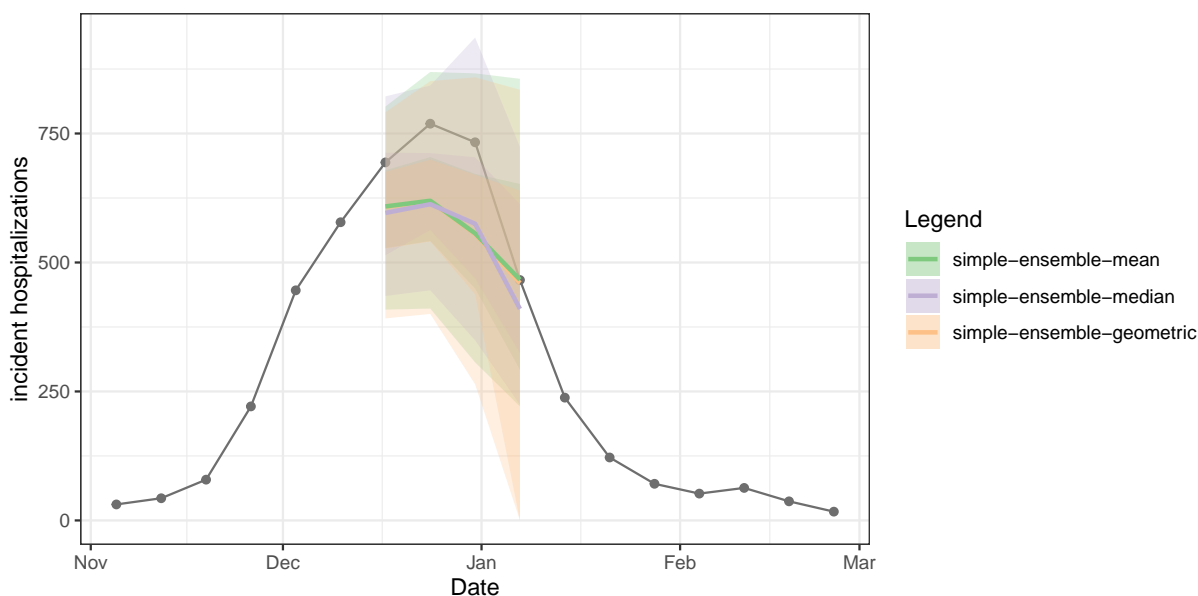
Figure 5: Three different ensembles for weekly incident influenza hospitalizations in Massachusetts. Each ensemble combines individual predictions from the example hub (Figure 3) using a different method: arithmetic mean, geometric mean, or median. All methods correspond to variations of the quantile 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.

4.2.2 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"
+    )
```

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, 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 6, 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.
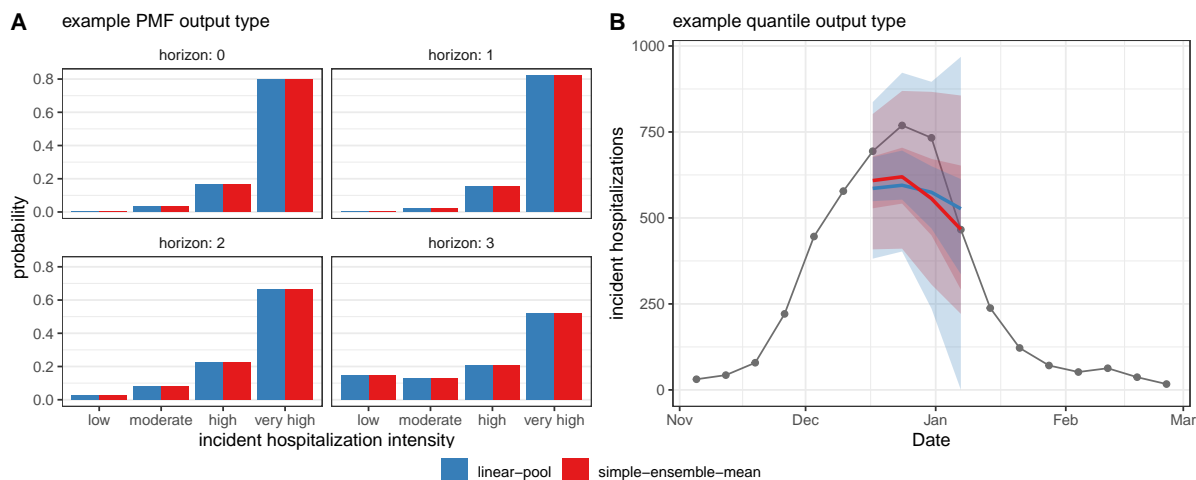
Figure 6: Comparison of results from `linear_pool()` (blue) and `simple_ensemble()` (red). (Panel A) Ensemble predictions of Massachusetts incident influenza hospitalization intensity (classified as low, moderate, high, or very high), which provide an example of PMF output type. (Panel B) Ensemble predictions 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).

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.)

```
> 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)[46]. 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()` defaults to create and equally-weighted ensemble by simply collecting and returning 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. 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.

When requesting a linear pool composed of a subset of the input sample predictions, the user must identify the task ID variables which together identify a single modeled unit. This group of independent task ID variables is called the compound task ID set and are specified using the `compound_taskid_set` parameter to ensure the subsetting of sample predictions is performed correctly. Samples summarizing a marginal distribution will have a compound task ID set made up of all the task ID variables, while samples summarizing a joint distribution will have a compound task ID set that contains only task ID variables that do not display dependence within their values.

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). 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.

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

5 Example: in-depth analysis of forecast data

To further demonstrate the differences between the two ensembling functions and the utility of the `hubEnsembles` package, we examine a more complex example. Unlike the previous section's basic showcase of functionality to create multi-model ensembles, we use this case study to provide a more complete analysis that compares and evaluates ensemble model performance using real forecasts collected by a modeling hub, with an overarching goal of choosing a single best ensembling approach for the application.

Since 2013, the US Centers for Disease Control and Prevention (CDC) has been soliciting forecasts of seasonal influenza from modeling teams through a collaborative challenge called FluSight[48]. We use a subset of these predictions to create four equally-weighted ensembles with `simple_ensemble()` and `linear_pool()` and compare the resulting ensembles' performance. The ensembling methods chosen for this case study consist of a quantile (arithmetic) mean, a quantile median, a linear pool with normal tails, and a linear pool with lognormal tails. Note that only a select portion of the code is shown in this manuscript for brevity, although 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). More specifically, the figures and tables supporting this analysis are generated reproducibly 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 begin by querying the component forecasts used to generate the four ensembles from Zoltar[49], a repository designed to archive forecasts created by the Reich Lab at UMass Amherst. For this analysis we only consider FluSight predictions in a quantile format from the 2021-2022 and 2022-2023 seasons. These forecasts were stored in two data objects, split by season, called `flu_forecasts-zoltar_21-22.rds` and `flu_forecasts-zoltar_22-23.rds`, and 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)
```

Table 8: An example prediction of weekly incident influenza hospitalizations pulled directly from Zoltar. 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.)

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

Forecasts must conform to hubverse standards to be fed into either of the ensembling functions, so we first transform the raw forecasts using the `as_model_out_tbl()`[1] function from the `hubUtils` package. Here, we specify the task ID variables `forecast_date` (when the forecast was made), `location`, `horizon`, and `target`.

```
> flu_forecasts_hubverse <- flu_forecasts_raw |>
+    dplyr::rename(forecast_date = timezero, location = unit) |>
+    tidyr::separate(target, sep = " ", convert = TRUE,
+                    into = c("horizon", "target"), extra = "merge") |>
+    dplyr::mutate(target_end_date =
+                    ceiling_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 = "-",
```

---

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

```
+     trim_to_task_ids = FALSE,
+     hub_con = NULL,
+     task_id_cols =
+       c("forecast_date", "location", "horizon", "target", target_end_date),
+     remove_empty = TRUE
+   )
```

Prior to ensemble calculation (shown later in this section), we filter out any predictions (defined by a unique combination of task ID variables) that did not include all 23 quantiles specified by FluSight ($\theta \in \{.010, 0.025, .050, .100, ..., .900, .950, .990\}$). The FluSight baseline and median ensemble models generated by the FluSight hub are also excluded from the component forecasts. We chose to remove the baseline to match the composition of models used to create the official FluSight ensemble.

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.

Table 9: An example prediction of weekly incident influenza hospitalizations. 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.

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

In both seasons, forecasts were made for the same locations (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 above), and target (week ahead incident flu 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 combine the component model outputs to generate predictions from each ensemble model. We begin by excluding the baseline model from the set of predictions that will be combined. Then, we create one object to store the ensemble results generated from each method we are interested in comparing.

```
> 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 evaluate the performance of these ensembles using scoring metrics that measure the accuracy and calibration of their forecasts. Here, we choose several common metrics in forecast evaluation, including mean absolute error (MAE), weighted interval score (WIS)[50], 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[50]. 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 weights these penalties across all prediction intervals provided. A lower WIS value indicates a more accurate forecast[50]. 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 compares to that of the FluSight baseline model. These metrics are calculated as

$$\text{rWIS} = \frac{\text{WIS}_{\text{model } m}}{\text{WIS}_{\text{baseline}}} \qquad\qquad \text{rMAE} = \frac{\text{MAE}_{\text{model } m}}{\text{MAE}_{\text{baseline}}},$$

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. 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 using the `score_forecasts()`[2] function from the `covidHubUtils` package, as a hubverse package for scoring and evaluation has not yet been fully implemented. This function outputs each of the metrics described above. We use median forecasts taken from the 0.5 quantile for the MAE evaluation.

5.2 Performance results across ensembles

The quantile median ensemble has the best overall performance in terms of WIS and MAE (and the relative versions of these metrics), and has coverage rates that were close to the nominal levels (Table 10). The two linear opinion pools have very similar performance to each other. These methods have the second-best performance as measured by WIS and MAE, but they have the highest 50% and 95% coverage rates, with empirical coverage that was well above the nominal coverage rate. The quantile mean performs the worst of the ensembles with the highest MAE, which is substantially different from that of the other ensembles.

---

[2]https://reichlab.io/covidHubUtils/reference/score_forecasts.html

Table 10: Summary of overall model performance across both seasons, averaged over all locations except the US national location and sorted by ascending WIS. The quantile median ensemble has the best value for every metric except 50% coverage rate, though metric values are often quite similar among the models.

model	wis	rwis	mae	rmae	cov50	cov95
median-ensemble	18.158	0.794	27.360	0.933	0.597	0.922
lp-normal	19.745	0.863	27.932	0.953	0.709	0.990
lp-lognormal	19.747	0.863	27.933	0.953	0.708	0.990
mean-ensemble	20.180	0.882	29.582	1.009	0.595	0.889
Flusight-baseline	22.876	1.000	29.315	1.000	0.604	0.881

Plots of the models' forecasts can aid our understanding about the origin of these accuracy differences. For example, the linear opinion pools consistently have some of the widest prediction intervals, and consequently the highest coverage rates. The median ensemble, which has the best WIS, balanced interval width with calibration best overall, with narrower intervals than the linear pools that still achieved near-nominal coverage on average across all time points. The quantile mean's interval widths vary, though it usually has narrower intervals than the linear pools. However, this model's point forecasts have a larger error margin compared to the other ensembles, especially at longer horizons. This pattern is demonstrated in Figure 7 for the 4-week ahead forecast in California following the 2022-23 season peak on December 5, 2022. Here, the quantile mean predicted a continued increase in hospitalizations, at a steeper slope than the other ensemble methods.

Averaging across all time points, the median model can be seen to have the best scores for every metric. It outperforms the mean ensemble by a similar amount for both MAE and WIS, particularly around local times of change (see Figure 8 and Figure 9). The median ensemble also has better coverage rates than the mean ensemble in the tails of the distribution (95% intervals, see Figure 10) and similar coverage in the center (50% intervals). The median model also outperforms the linear pools for most weeks, with the greatest differences in scores being for WIS and coverage rates (Figure 9 and Figure 10). This seems to indicate that the linear pools' estimates are usually too conservative, with their wide intervals and higher-than-nominal coverage rates being penalized by WIS. However, during the 2022-2023 season there are several localized times when the linear pools showcased better one-week-ahead forecasts than the median ensemble (Figure 9). These localized instances are characterized by similar MAE values (Figure 9) for the two methods and poor median ensemble coverage rates (Figure 10). In these instances, the wide intervals from the linear pools were useful in capturing the eventually-observed hospitalizations, usually during times of rapid change.

All of the ensemble variations outperform the baseline model in this analysis, led by the quantile median which had the best scores overall for WIS, MAE, 50% PI coverage, and 95% PI coverage. However, other models sometimes demonstrated better performance, especially

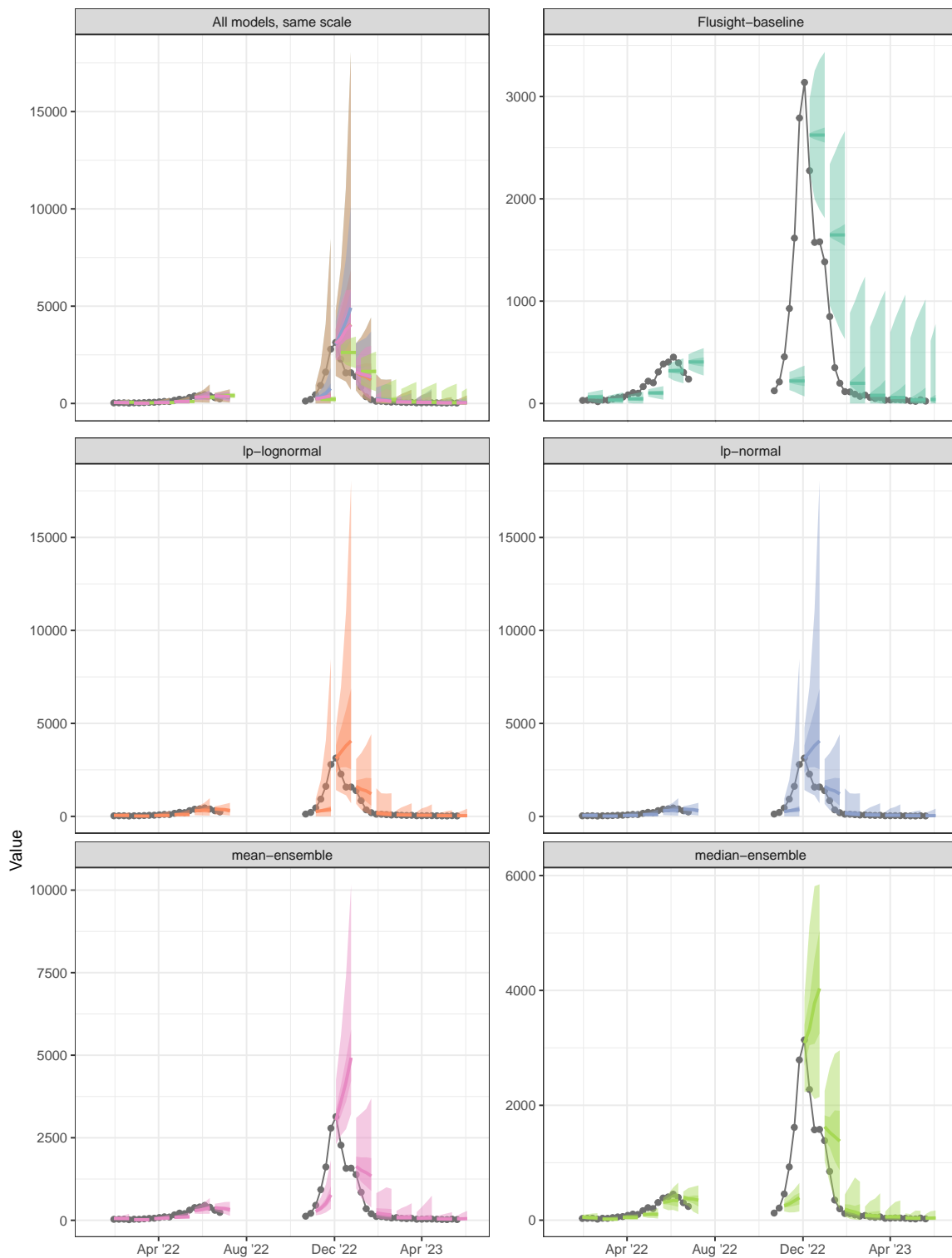Weekly Incident Hospitalizations for Influenza in California

Figure 7: One to four week ahead forecasts for select dates plotted against target data for California. The first panel shows all models on the same scale. All other panels show forecasts for each individual model, with varying y-axis scales, and their prediction accuracy as compared to observed influenza hospitalizations.
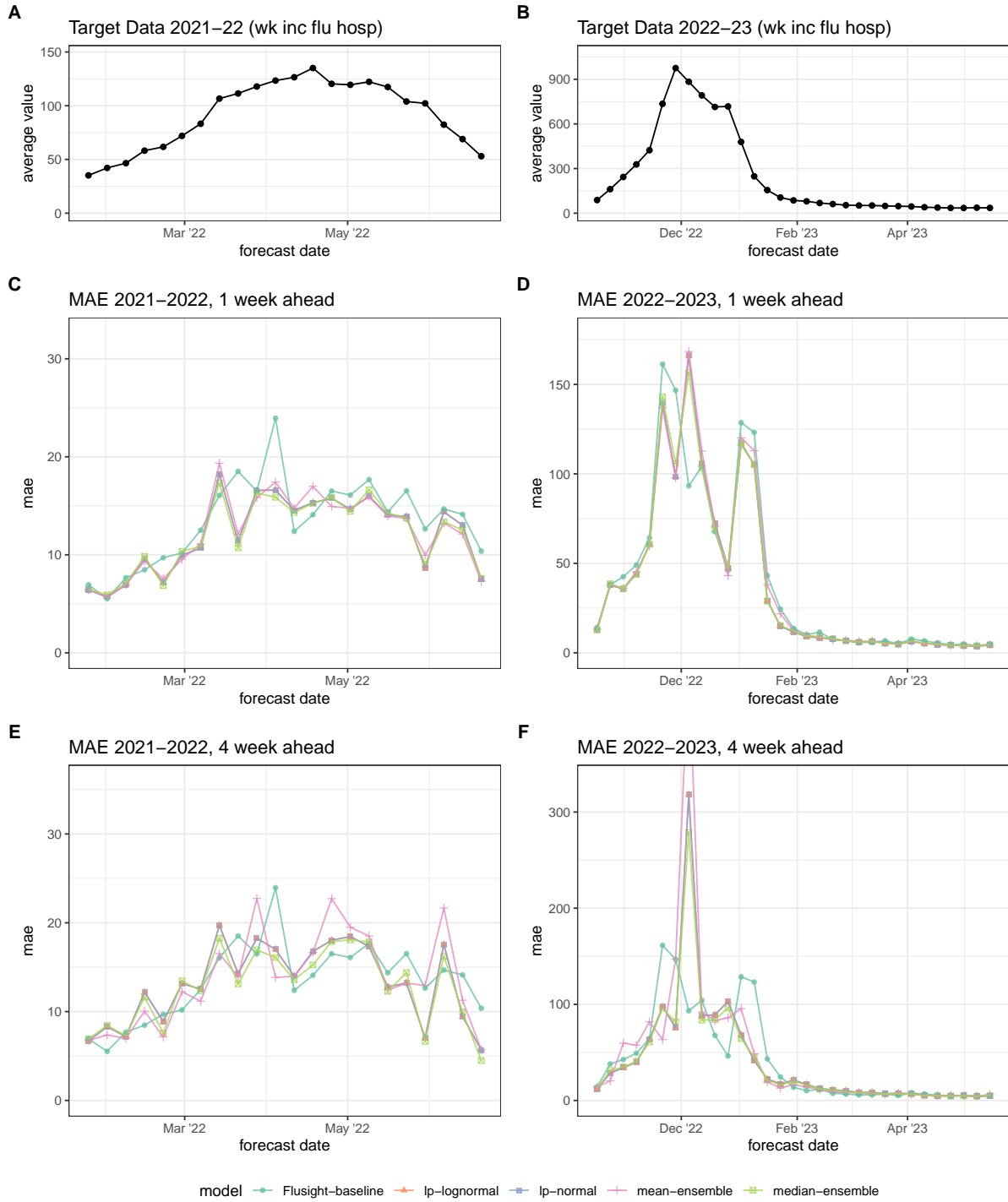
Figure 8: Mean absolute error (MAE) averaged across all locations. Average target data across all locations for 2021-2022 (A) and 2022-2023 (B) seasons for reference. For each season, average MAE is shown for 1-week (C-D) and 4-week ahead (E-F) forecasts. Results are plotted for each ensemble model (colors) across the entire season. Lower values indicate better performance.

33
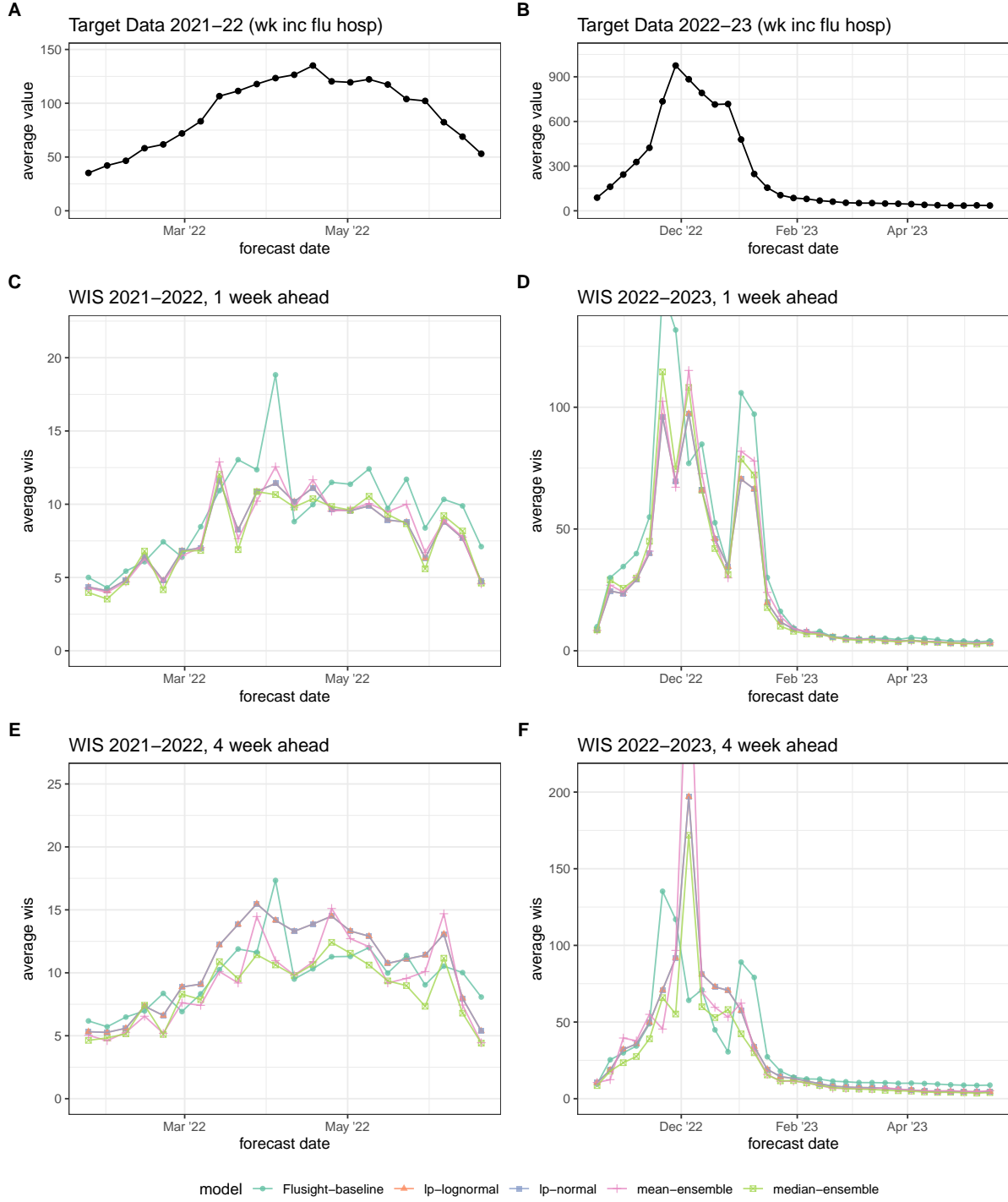
Figure 9: Weighted interval score (WIS) averaged across all locations. Average target data across all locations for 2021-2022 (A) and 2022-2023 (B) seasons for reference. For each season, average WIS is shown for 1-week (C-D) and 4-week ahead (E-F) forecasts. Results are plotted for each ensemble model (colors) across the entire season. Lower values indicate better performance.

34
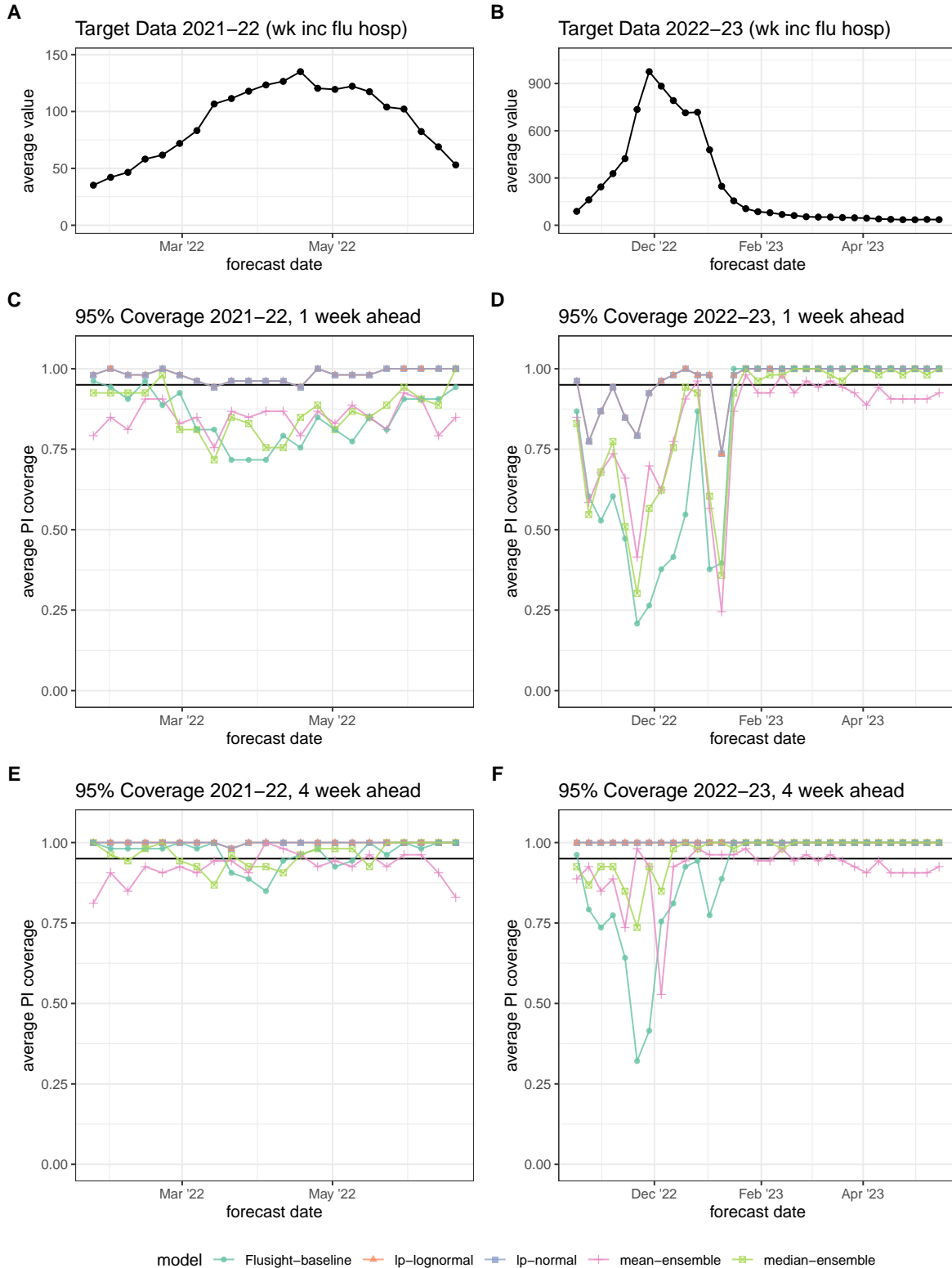
Figure 10: 95% prediction interval (PI) coverage averaged across all locations. Average target data across all locations for 2021-2022 (A) and 2022-2023 (B) seasons for reference. For each season, average coverage is shown for 1-week (C-D) and 4-week ahead (E-F) forecasts. Results are plotted for each ensemble model (colors) across the entire season. Ideal coverage of 95% is shown (black horizontal line); values closer to 95% indicate better performance.

around the season's peak. This seems to indicate that the quantile median may not necessarily be the top-performing model for every influenza season.

The choice of an appropriate ensemble aggregation method may 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 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 biological settings. 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 generate and communicate ensemble results[10,35]. 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[23]. Our case study on seasonal influenza forecasts in the US demonstrates this point. The quantile median ensemble performs best overall for a range of metrics, including weighted interval score, mean absolute error, and prediction interval coverage. Yet, the other ensemble methods we tested also showcased a clear improvement over the baseline model and even outperformed the quantile median for certain weeks, particularly during periods of rapid change when outlying component forecasts are likely more important. The accuracy gains from ensemble models motivate the use of a "hub-based" approach to prediction for infectious diseases, public health, and in other fields.

Acknowledgements

Consortium of Infectious Disease Modeling Hubs

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 Hygene and Tropical Medicine), Shaun Truelove (Johns Hopkins University), Martha Zorn (University of Massachusetts Amherst)

References

1. Clemen RT. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting.* 1989;5(4):559-583. doi:10.1016/0169-2070(89)90012-5

2. Timmermann A. Chapter 4 Forecast Combinations. In: Vol 1. Elsevier; 2006:135-196. doi:10.1016/S1574-0706(05)01004-9

3. Hibon M, Evgeniou T. To combine or not to combine: selecting among forecasts and their combinations. *International Journal of Forecasting.* 2005;21(1):15-24. doi:10.1016/j.ijforecast.2004.05.002

4.   Alley RB, Emanuel KA, Zhang F. Advances in weather prediction. *Science.* 2019;363(6425):342-344. doi:10.1126/science.aav7274

5.   Tebaldi C, Knutti R. The use of the multi-model ensemble in probabilistic climate projections. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences.* 2007;365(1857):2053-2075. doi:10.1098/rsta.2007.2076

6.   Aastveit KA, Mitchell J, Ravazzolo F, Dijk HK van. The Evolution of Forecast Density Combinations in Economics. *Tinbergen Institute Discussion Papers.* Published online 2018. https://hdl.handle.net/10419/185588

7.   Shaman J, Kandula S. Improved Discrimination of Influenza Forecast Accuracy Using Consecutive Predictions. *PLoS Currents.* 2015;7:ecurrents.outbreaks.8a6a3df285af7ca973fab4b22e10911e. doi:10.1371/currents.outbreaks.8a6a3df285af7ca973fab4b22e10911e

8.   Held L, Meyer S, Bracher J. Probabilistic forecasting in infectious disease epidemiology: The 13th Armitage lecture. *Statistics in Medicine.* 2017;36(22):3443-3460. doi:10.1002/sim.7363

9.   Ray EL, Sakrejda K, Lauer SA, Johansson MA, Reich NG. Infectious disease prediction with kernel conditional density estimation. *Statistics in Medicine.* 2017;36(30):4908-4929. doi:10.1002/sim.7488

10.   Reich NG, Lessler J, Funk S, et al. Collaborative hubs: Making the most of predictive epidemic modeling. *American Journal of Public Health.* 2022;112(6):839-842. doi:10.2105/AJPH.2022.306831

11.   McGowan CJ, Biggerstaff M, Johansson M, et al. Collaborative efforts to forecast seasonal influenza in the United States, 2015–2016. *Scientific Reports.* 2019;9(1):683. doi:10.1038/s41598-018-36361-9

12.   Reich NG, McGowan CJ, Yamana TK, et al. Accuracy of real-time multi-model ensemble forecasts for seasonal influenza in the U.S. *PLOS computational biology.* 2019;15(11):e1007486. doi:10.1371/journal.pcbi.1007486

13.   Johansson MA, Apfeldorf KM, Dobson S, et al. An open challenge to advance probabilistic forecasting for dengue epidemics. *Proceedings of the National Academy of Sciences.* 2019;116(48):24268-24274. doi:10.1073/pnas.1909865116

14. Holcomb KM, Mathis S, Staples JE, et al. Evaluation of an open forecasting challenge to assess skill of West Nile virus neuroinvasive disease prediction. *Parasites & Vectors.* 2023;16(1):11. doi:10.1186/s13071-022-05630-y

15. Cramer EY, Ray EL, Lopez VK, et al. Evaluation of individual and ensemble probabilistic forecasts of COVID-19 mortality in the united states. *Proceedings of the National Academy of Sciences.* 2022;119(15):e2113561119. doi:10.1073/pnas.2113561119

16. Sherratt K, Gruson H, Grah R, et al. Predictive performance of multi-model ensemble forecasts of COVID-19 across European nations. *eLife.* 2023;12:e81916. doi:10.7554/eLife.81916

17. Howerton E, Contamin L, Mullany LC, et al. Evaluation of the US COVID-19 Scenario Modeling Hub for informing pandemic response under uncertainty. *Nature Communications.* 2023;14(1):7260. doi:10.1038/s41467-023-42680-x

18. Yamana TK, Kandula S, Shaman J. Superensemble forecasts of dengue outbreaks. *Journal of The Royal Society Interface.* 2016;13(123):20160410. doi:10.1098/rsif.2016.0410

19. Ray EL, Reich NG. Prediction of infectious disease epidemics via weighted density ensembles. *PLOS computational biology.* 2018;14(2):e1005910. doi:10.1371/journal.pcbi.1005910

20. Viboud C, Sun K, Gaffey R, et al. The RAPIDD ebola forecasting challenge: Synthesis and lessons learnt. *Epidemics.* 2018;22:13-21. doi:10.1016/j.epidem.2017.08.002

21. Colón-González FJ, Bastos LS, Hofmann B, et al. Probabilistic seasonal dengue forecasting in Vietnam: A modelling study using superensembles. *PLOS Medicine.* 2021;18(3):e1003542. doi:10.1371/journal.pmed.1003542

22. Paireau J, Andronico A, Hozé N, 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.* 2022;119(18):e2103302119. doi:10.1073/pnas.2103302119

23. Howerton E, Runge MC, Bogich TL, et al. Context-dependent representation of within- and between-model uncertainty: Aggregating probabilistic predictions in infectious disease epidemiology. *Journal of The Royal Society Interface.* 2023;20(198):20220659. doi:10.1098/rsif.2022.0659

24.   Ray EL, Brooks LC, Bien J, et al. Comparing trained and untrained probabilistic ensemble forecasts of COVID-19 cases and deaths in the United States. *International Journal of Forecasting.* 2023;39(3):1366-1383. doi:10.1016/j.ijforecast.2022.06.005

25.   Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research.* 2011;12(85):2825-2830. doi:10.5555/1953048.2078195

26.   Weiss Christoph,E, Raviv E, Roetzer G. Forecast Combinations in R using the ForecastComb Package. *The R Journal.* 2019;10(2):262. doi:10.32614/RJ-2018-052

27.   Bosse N, Yao Y, Abbott S, Funk S. *Stackr: Create Mixture Models From Predictive Samples.*; 2023.

28.   Couch S, Kuhn M. *Stacks: Tidy Model Stacking.*; 2023.

29.   Consortium of Infectious Disease Modeling Hubs. The hubverse: Open tools for collaborative forecasting. Published online 2025. https://hubverse.io/en/latest/index.html

30.   Vincent SB. *The Function of the Vibrissae in the Behavior of the White Rat.* PhD thesis. University of Chicago; 1912.

31.   Stone M. The opinion pool. *The Annals of Mathematical Statistics.* 1961;32(4):1339-1342.

32.   Lichtendahl KC, Grushka-Cockayne Y, Winkler RL. Is it better to average probabilities or quantiles? *Management Science.* 2013;59(7):1594-1611. doi:10.1287/mnsc.1120.1667

33.   Hollingsworth TD, Medley GF. Learning from multi-model comparisons: Collaboration leads to insights, but limitations remain. *Epidemics.* 2017;18:1-3. doi:10.1016/j.epidem.2017.02.014

34.   Shea K, Runge MC, Pannell D, et al. Harnessing multiple models for outbreak management. *Science.* 2020;368(6491):577-579. doi:10.1126/science.abb9934

35.   Borchering RK, Healy JM, Cadwell BL, et al. Public health impact of the U.S. Scenario Modeling Hub. *Epidemics.* 2023;44:100705. doi:10.1016/j.epidem.2023.100705

36.   Reich NG, Brooks LC, Fox SJ, et al. A collaborative multiyear, multimodel assessment of seasonal influenza forecasting in the United States. *Proceedings of the National Academy of Sciences.* 2019;116(8):3146-3154. doi:10.1073/pnas.2200703119

37. Borchering RK, Mullany LC, Howerton E, 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. *The Lancet Regional Health – Americas.* 2023;17:100398. doi:10.1016/j.lana.2022.100398

38. Loo SL, Howerton E, Contamin L, et al. The US COVID-19 and Influenza Scenario Modeling Hubs: Delivering long-term projections to guide policy. *Epidemics.* 2024;46:100738. doi:10.1016/j.epidem.2023.100738

39. Jung S, Loo SL, Howerton E, et al. Potential impact of annual vaccination with reformulated COVID-19 vaccines: Lessons from the US COVID-19 scenario modeling hub. *PLOS Medicine.* 2024;21(4):e1004387. doi:10.1371/journal.pmed.1004387

40. Dean NE, Pastore y Piontti A, Madewell ZJ, et al. Ensemble forecast modeling for the design of COVID-19 vaccine efficacy trials. *Vaccine.* 2020;38(46):7213-7216. doi:10.1016/j.vaccine.2020.09.031

41. Pollett S, Johansson MA, Reich NG, et al. Recommended reporting items for epidemic forecasting and prediction research: The EPIFORGE 2020 guidelines. *PLOS Medicine.* 2021;18(10):e1003793. doi:10.1371/journal.pmed.1003793

42. Oidtman RJ, Omodei E, Kraemer MUG, et al. Trade-offs between individual and ensemble forecasts of an emerging infectious disease. *Nature Communications.* 2021;12(1):5379. doi:10.1038/s41467-021-25695-0

43. Wattanachit N, Ray EL, McAndrew TC, Reich NG. Comparison of combination methods to create calibrated ensemble forecasts for seasonal influenza in the U.S. *Statistics in Medicine.* 2023;42(26):4696-4712. doi:10.1002/sim.9884

44. McAndrew T, Reich NG. Adaptively stacking ensembles for influenza forecasting. *Statistics in Medicine.* 2021;40(30):6931-6952. doi:10.1002/sim.9219

45. Fox SJ, Kim M, Meyers LA, Reich NG, Ray EL. Optimizing Disease Outbreak Forecast Ensembles. 2024;30(9):1967-1969. doi:10.3201/eid3009.240026

46. Ray EL, Gerding A. *Distfromq: Reconstruct a Distribution from a Collection of Quantiles.*; 2024.

47. Niederreiter H. *Random Number Generation and Quasi-Monte Carlo Methods.* Society for Industrial; Applied Mathematics; 1992.

48. CDC. About flu forecasting. Published online 2023. https://www.cdc.gov/flu/weekly/flusight/how-flu-forecasting.htm

49. Reich NG, Cornell M, Ray EL, House K, Le K. The Zoltar forecast archive, a tool to standardize and store interdisciplinary prediction research. *Scientific Data.* 2021;8(1):59. doi:10.1038/s41597-021-00839-5

50. Bracher J, Ray EL, Gneiting T, Reich NG. Evaluating epidemic forecasts in an interval format. *PLOS Computational Biology.* 2021;17(2):e1008618. doi:10.1371/journal.pcbi.1008618