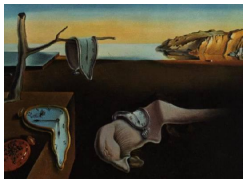

Journal of Statistical Software

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

modelimportance: Evaluating model importance within a multi-model ensemble in R

Minsu Kim
University of Massachusetts Amherst

Li Shandross
University of Massachusetts Amherst

Evan L. Ray
University of Massachusetts Amherst
CVS Health

Nicholas G. Reich
University of Massachusetts Amherst

Abstract

Ensemble forecasts are commonly used to support decision-making and policy planning across various fields because they often offer improved accuracy and stability compared to individual models. As each model has its own unique characteristics, understanding and measuring the value each constituent model adds to the overall accuracy of the ensemble can support the construction of effective ensembles. The R package **modelimportance** provides tools to quantify how each component model contributes to the accuracy of ensemble performance for both point and probabilistic forecasts. It supports multiple functionalities; it allows users to specify which ensemble approach to implement and which model importance metric to use. Additionally, the software offers customizable options for handling missing values. These features enable the package to serve as a versatile tool for researchers and practitioners. It helps not only in constructing an effective ensemble model across a wide range of forecasting tasks, but also in understanding the role of each model within the ensemble and gaining insights into individual models themselves. This package follows the 'hubverse' framework, which is a collection of open-source software and tools developed to promote collaborative modeling hub efforts and simplify their setup and operation. Doing so enables seamless integration and flexibility with other forecasting tools and systems, allowing many analyses to be performed on existing and ongoing hubs.

Keywords: ensemble, forecast, prediction, model importance, Shapley value, R.

1. Introduction

Ensemble forecasting is a method to produce a single, consolidated prediction by combining

forecasts generated from different models. While each model's strengths are pronounced, its weaknesses are counterbalanced, which leads to an ensemble forecast that is more robust and accurate (Gneiting and Raftery 2005; Hastie *et al.* 2001). Specifically, ensembles effectively mitigate the bias and variance arising from the predictions of individual models by averaging them out, and aggregating in this way can reduce prediction errors and improve overall performance. Enhanced prediction accuracy and robustness enable the achievement of more reliable predictions, thereby improving decision-making. For this reason, ensemble forecasting is widely used across various domains such as weather forecasting (Jordan A. Guerra *et al.* 2020; Gneiting and Raftery 2005), financial modeling (Sun *et al.* 2020; He *et al.* 2023), and infectious disease outbreak forecasting (Ray and Reich 2018; Reich *et al.* 2019; Lutz *et al.* 2019; Viboud *et al.* 2018). For example, throughout the COVID-19 pandemic, the US COVID-19 Forecast Hub collected individual models developed by over 90 different research groups and built a probabilistic ensemble forecasting model for COVID-19 cases, hospitalizations, and deaths in the US based on those models' predictions, which served as the official short-term forecasts for the US Centers for Disease Control and Prevention (CDC) (Cramer *et al.* 2022).

The quality of forecasts is assessed by evaluating their error, bias, sharpness, and/or calibration using different scoring metrics. The selection of the scoring metrics depends on the type of forecast: point forecasts (e.g., mean, median) and probabilistic forecasts (e.g., quantiles, samples, predictive cumulative distribution functions, probability mass function). Commonly used assessment tools for point forecasts are the mean absolute error (MAE) and the mean squared error (MSE), which calculate the average magnitude of forecast errors. Scoring metrics for probabilistic forecasts consider the uncertainty and variability in predictions and provide concise evaluations through numerical scores (Gneiting and Raftery 2007). Some examples include the weighted interval score (WIS) for the quantile-based forecasts and the continuous ranked probability score (CRPS) for the forecasts taking the form of predictive cumulative distribution functions (Bracher *et al.* 2021). We note that CRPS is a general scoring rule that can be computed either analytically in closed form or numerically from samples, and WIS is a quantile-based approximation of CRPS.

Several R packages have been developed for this purpose. To name a few, the **fable** package (O'Hara-Wild *et al.* 2024) is widely used for univariate time series forecasting and includes functions for accuracy measurement. The **Metrics** (Hamner and Frasco 2018) and **MLmetrics** (Yan 2024) provide a wide range of performance metrics specifically designed for evaluating machine learning models. The **scoringRules** (Jordan *et al.* 2019) package offers a comprehensive set of proper scoring rules for evaluating probabilistic forecasts and supports both univariate and multivariate settings. The **scoringutils** (Bosse *et al.* 2022) package offers additional features to the functionality provided by **scoringRules**, which makes it more useful for certain tasks, such as summarizing, comparing, and visualizing forecast performance. These packages have been valuable to evaluate individual models as independent entities, using performance metrics selected for each specific situation or problem type. However, they do not measure the individual models' contributions to the enhanced predictive accuracy when used as part of an ensemble. Kim *et al.* (2024) demonstrate that a model's individual performance does not necessarily correspond to its contribution as a component within an ensemble. Our developed package introduces this capability. The **modelimportance** package provides tools to evaluate the role of each model as an ensemble member within an ensemble model, rather than focusing on the individual predictive performance per se.

In ensemble forecasting, certain models contribute more significantly to the overall predictions than others. Assessing the impact of each component model on ensemble predictions is methodologically similar to determining variable importance in traditional regression and machine learning models, where variable importance measures evaluate how much individual variables decrease in accuracy of the model's predictive performance or reduce the average loss. R packages such as **randomForest** (Liaw and Wiener 2002), **caret** (Kuhn 2008), **xgboost** (Chen *et al.* 2024), and **gbm** (Ridgeway and Developers 2024) implement these functions for different types of models: random forest models, general machine learning models, extreme gradient boosting models, and generalized boosted regression models, respectively. These packages focus on feature-level importance within a single model and do not measure the contribution of individual models within an ensemble. The **modelimportance** package addresses this limitation. The tools in the **modelimportance** quantify how each component model helps enhance the ensemble model's predictive performance. They assign numerical scores to each model based on a selected metric that measures forecast accuracy, depending on the forecast type.

These capabilities provide unique support for hub organizers who refer to the entity responsible for launching and managing a collective modeling hub. They coordinate multiple teams to produce forecasts and integrate their predictions into an ensemble forecast (Shandross *et al.* 2024), which, as mentioned earlier, is known for having better performance compared to individual models. Examples include the US CDC and the European Centre for Disease Prevention and Control. The **modelimportance** package can even strengthen the benefits of a multi-model ensemble by helping these organizations create more effective ensemble forecasts based on the precise evaluation of each model's contribution. Specifically, **modelimportance** follows 'hubverse' standards, where 'hubverse' offers a set of publicly available software and data tools developed to promote collaborative modeling hub efforts and reduce the effort required to set up and operate them (Consortium of Infectious Disease Modeling Hubs 2024). Adherence to the model output formats specified by the hubverse convention enables many analyses to be performed on existing and ongoing hubs by seamless integration and flexibility with other forecasting tools and systems. We note that there are over a dozen active hubs running as of fall 2025, with more in planning stages.

We highlight some strong development practices we employed, such as unit testing of individual functions, continuous integration testing on different operating systems, and independent code review. This emphasis on quality control is a key strength of this work and distinguishes it from other academic software development projects.

The paper proceeds as follows. Section 2 describes how the **modelimportance** package relates to the hubverse framework, including its dependencies, the model output formats defined within hubverse, and the structure of data presentation for both forecasts and actual observations. Section 3 presents two algorithms implemented in **modelimportance** for calculating the model importance metric: leave-one-model-out and leave-all-subsets-of-models-out. We demonstrate the various functionalities **modelimportance** supports in Section 4 and highlight our quality assurance measures and its open access in Section 5. Some examples are provided in Section 6. We close this paper with concluding remarks and a discussion of possible extensions.

2. Data

2.1. Relationship and dependencies on hubverse

The **modelimportance** package is designed to work with the hubverse framework and, accordingly, depends on several packages in the hubverse ecosystem, such as **hubUtils** (Krystalli and Shandross (2025)), **hubEnsembles** (Shandross *et al.* (2025)), **hubEvals** (Reich *et al.* (2025)), and **hubExamples** (Ray *et al.* (2025)). **modelimportance** uses a `model_out_tbl` S3 class as the model output format defined in **hubUtils**, which consists of utility functions to standardize prediction files and data formats (details in Section 2.2). Ensembling predictions from multiple models relies on **hubEnsembles**, which offers a broadly applicable framework to construct multi-model ensembles using various ensemble methods. Calculation of forecast accuracy using various metrics is based on **hubEvals**, which internally leverages **scoringutils**. We use the example datasets from **hubExamples** for testing and demonstration purposes (see Section 6).

2.2. Model output format

Model outputs are structured in a tabular format designed specifically for predictions, which is a formal S3 object called `model_out_tbl`. In the hubverse standard, each row represents an individual prediction or a component of a prediction for a single task, and its details are described in multiple columns through which one can identify the unique label assigned to each forecasting model, task characteristics, prediction representation type, and predicted values (Shandross *et al.* 2024). To elaborate on the task characteristics, each prediction task means a specific forecasting problem and it can be described by a set of task ID variables. Examples of such variables include a date on which forecasts are generated, the target to predict (e.g., flu-related incident deaths, cases, or hospitalizations), and the prediction horizon, which is the length of time into the future from the point when a model generate its forecast, for a specific location on a certain target date. Table 1 illustrates short-term forecasts of weekly incident influenza hospitalizations in the US for Massachusetts, generated by the model 'Flusight-baseline' on December 17, 2022, in the `model_out_tbl` format. The `model_id` column lists a uniquely identified model name. All of the `reference_date`, `target`, `horizon`, `location`, and `target_end_date` columns are all referred to as the task ID variables, which together defines the task characteristics. Note that the forecast generation date and the target date for which the prediction is made are mapped to the `reference_date` and `target_end_date` columns, respectively, and the location is represented based on the FIPS code (e.g., '25' for Massachusetts). The time length to the `target_end_date`, which is the number of weeks ahead from the `reference_date`, is indicated in the `horizon` column. The prediction representation is specified as 'quantile' in the `output_type` column, and details are represented in the `output_type_id` column with seven quantiles of 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, and 0.95 for each target end date. The predicted value corresponding to each quantile is recorded in the `value` column.

Figure 1 visualize the information on the prediction task provided by Table 1 for three models. For each model, the quantile-based forecasts are shown for the target end dates of December 24, 2022 (horizon 1), December 31, 2022 (horizon 2), and January 07, 2023 (horizon 3), which were made on December 17, 2022 based on the historical data available as of that date. The

model_id	reference_date	target	horizon	location	target_end_date	output_type	output_type_id	value
Flusight-baseline	2022-12-17	wk inc flu hosp	1	25	2022-12-24	quantile	0.05	496
Flusight-baseline	2022-12-17	wk inc flu hosp	1	25	2022-12-24	quantile	0.1	536
Flusight-baseline	2022-12-17	wk inc flu hosp	1	25	2022-12-24	quantile	0.25	566
Flusight-baseline	2022-12-17	wk inc flu hosp	1	25	2022-12-24	quantile	0.5	582
Flusight-baseline	2022-12-17	wk inc flu hosp	1	25	2022-12-24	quantile	0.75	598
Flusight-baseline	2022-12-17	wk inc flu hosp	1	25	2022-12-24	quantile	0.9	629
Flusight-baseline	2022-12-17	wk inc flu hosp	1	25	2022-12-24	quantile	0.95	668
Flusight-baseline	2022-12-17	wk inc flu hosp	2	25	2022-12-31	quantile	0.05	454
Flusight-baseline	2022-12-17	wk inc flu hosp	2	25	2022-12-31	quantile	0.1	518
Flusight-baseline	2022-12-17	wk inc flu hosp	2	25	2022-12-31	quantile	0.25	558

Table 1: Example of the model output for incident influenza hospitalizations (top 10 rows) extracted from `forecast_outputs` data in the **hubExamples** package.

prediction intervals defined by the lowest and highest quantiles (0.05 and 0.95) represent the uncertainty of the predictions. To give brief details in the interpretation, the Flusight-baseline model under-predicted the outcomes for the first two target dates (horizon 1 and 2), but it over-predicted the outcome for the last target date (horizon 3). Its prediction intervals are narrow compared to the other two models, which indicates that it is more confident about its predictions. However, two of three prediction intervals (horizons 1 and 2) failed to cover the eventually observed values, implying that the model was apparently overconfident.

2.3. Forecast data representation

Generally, quantitative forecasts can be categorized into either point forecasts or probabilistic forecasts. For a specific prediction task, point forecasts, represented by a single predicted value, provide a clear and concise prediction, making them easy to interpret and communicate. Probabilistic forecasts, on the other hand, provide a probability distribution over possible future values, which inherently involves uncertainty. They are represented in various ways, such as probability mass functions (pmf), cumulative distribution functions (cdf), samples, or probability quantiles (or intervals).

The `output_type` and `output_type_id` columns in the model output format, as defined by the hubverse convention, specify the forecast structure. Only one `output_type` is allowed, and it must be one of the 'mean', 'median', 'quantile', or 'pmf' in the **modelimportance** package: 'mean' or 'median' for point forecasts and 'quantile' or 'pmf' for probabilistic forecasts. As aforementioned, `output_type_id` column identifies addtional detailed information, such as specific quantile levels (e.g., "0.1", "0.25", "0.5", "0.75", and "0.9") for the 'quantile' output type and categorical values (e.g., "low", "moderate", "high", and "very high") for the 'pmf' output type. The predicted values for `pmf` are constrained to be between 0 and 1, indicating the probability at each categorical level, while they are unbounded numeric otherwise. Different output types correspond to different scoring rules for evaluating a model's prediction performance. Table 2 presents the output types and their associated scoring rules supported by the **modelimportance** package.

2.4. Oracle output data

The `oracle_output_data` is a data frame that contains the ground truth values for the

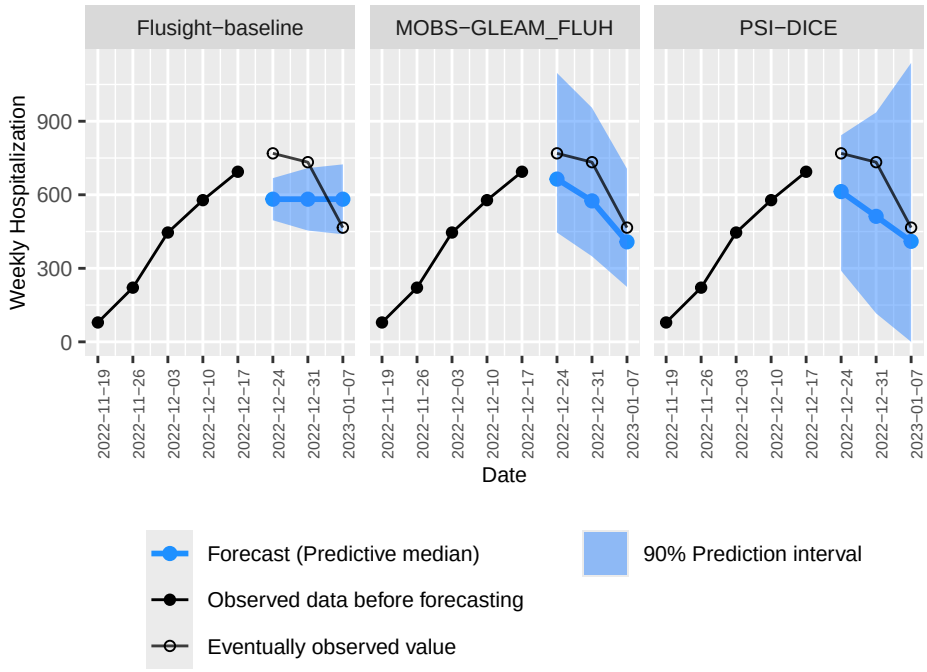

Figure 1: Example plot of three distributional forecasts corresponding to the model output for incident influenza hospitalizations shown in Table 1. Solid black dots indicate historically available data as of the forecast generation date, and open black circles indicate the eventually observed values. The blue dots represent predictive medians and the blue shaded area represents the corresponding 90% prediction interval defined by the 0.05 and 0.95 quantiles.

variables used to define modeling targets. It is referred to as "oracle" because it is formatted as if an oracle made a perfect point prediction equal to the truth. This data must follow the oracle output format defined in the hubverse standard, which includes independent task ID columns (e.g., `location`, `target_date`), the `output_type` column specifying the output type of the predictions and an `oracle_value` column for the observed values. As in the forecast data, if the `output_type` is either `"quantile"` or `"pmf"`, the `output_type_id` column is often required to provide further identifying information.

The `model_out_tbl` and `oracle_output_data` must have the same task ID columns and `output_type`, including `output_type_id` if necessary, which are used to match the predictions with the ground truth data.

3. Algorithms

This section provides a brief description of the leave one model out (LOMO) and leave all subsets of models out (LASOMO) algorithms, which are used to compute the model importance score. The basic idea of measuring the importance of each component model is to evaluate the change in ensemble performance when that model is included or excluded in the ensemble construction. More specifically, we compare the performance of an ensemble with

Output Type	Scoring Rule	Description
mean	RSE	Evaluate using the root squared error (RSE)
median	AE	Evaluate using the absolute error (AE)
quantile	WIS	Evaluate using the weighted interval score (WIS)
pmf	Log Score	Evaluate using the logarithm of the probability assigned to the true outcome (LogScore)

Table 2: Pairs of output types and their associated scoring rules for evaluating prediction performance.

and without a specific model for a specific task, and consider the difference in performance as the importance of that model for that task. We apply this idea to many tasks and aggregate the importance scores for that model across all tasks using averages. (Details can be found in Kim *et al.* (2024).)

LOMO involves creating an ensemble by excluding one component model from the entire set of models. Let A be a set of n models and F^i be a forecast produced by model i, where $i = 1, 2, \ldots, n$. Each ensemble excludes exactly one model while including all the others. $F^{A^{-i}}$ denotes the ensemble forecast constructed using all forecasts F^A except F^i. Model i's importance score using LOMO is calculated as the difference in accuracy, as measured by a specific score, between $F^{A^{-i}}$ and F^A (Algorithm 1). For example, when evaluating model 1 within an ensemble of three models ($n = 3$), LOMO creates an ensemble forecast $F^{\{2,3\}}$ using only F^2 and F^3. The performance of this reduced ensemble is then compared to the full ensemble forecast $F^{\{1,2,3\}}$, which incorporates all three models. We note that a model can make an ensemble better or worse, and thus the importance score for model 1 can be positive or negative accordingly.

Algorithm 1 Importance score calculation for one prediction task using leave one model out (LOMO) algorithm

Input:

- Set of n individual models, $A = \{1, 2, \ldots, n\}$, and their forecasts $\{F^1, F^2, ..., F^n\}$
- Truth value, y
- Function g to build an ensemble
- Scoring rule ψ to evaluate forecast skills

Output: importance metric of model i, $\phi^{i,\text{lomo}}$

1: Create an ensemble forecast F^A using g: $F^A \leftarrow g(\{F^1, F^2, ..., F^n\})$
2: Evaluate F^A using ψ: $\psi(F^A, y)$.
3: **for** each $i, (i = 1, 2, ..., n)$ **do**
4: $\quad F^{A^{-i}} \leftarrow g(\{F^j | \, j \neq i, j \in A\})$
5: $\quad$ Compute $\psi(F^{A^{-i}}, y)$.
6: $\quad \phi^{i,\text{lomo}} \leftarrow \psi(F^{A^{-i}}, y) - \psi(F^A, y)$
7: **end for**

Algorithm 2 Importance score calculation for one prediction task using leave all subsets of models out (LASOMO) algorithm

Input:

- Set of n individual models, $A = \{1, 2, \dots, n\}$, and their forecasts $\{F^1, F^2, ..., F^n\}$
- Truth value, y
- Function g to build an ensemble
- Scoring rule ψ to evaluate forecast skills

Output: Importance metric of model i, $\phi^{i,\text{lasomo}}$

for $i = 1$ to n **do**
  $\phi^{i,\text{lasomo}} \leftarrow 0$
  Make a list of non-empty subsets of A that does not contain i: $S_1, S_2, \cdots, S_{2^{n-1}-1}$
  for $j = 1$ to $2^{n-1} - 1$ **do**
    Assign a weight to the subset S_j:
    if all subsets have uniform weights **then**
      $\gamma_{S_j} \leftarrow \dfrac{1}{2^{n-1}-1}$
    else (*subset's weight depends on its size*)
      $\gamma_{S_j} \leftarrow \dfrac{1}{(n-1)\binom{n-1}{|S_j|}}$
    end if
    $F^{S_j} \leftarrow g(\{F^j | \, j \in S_j\})$
    $F^{S_j \cup \{i\}} \leftarrow g(\{F^i, F^j | \, j \in S_j\})$
    Compute $\psi(F^{S_j}, y)$ and $\psi(F^{S_j \cup \{i\}}, y)$
    $\phi^{i,\text{lasomo}} \leftarrow \phi^{i,\text{lasomo}} + \gamma_{S_j} \times \left[\psi(F^{S_j}, y) - \psi(F^{S_j \cup \{i\}}, y)\right]$
  end for
end for

On the other hand, LASOMO involves ensemble constructions from all possible subsets of models. For each subset S that does not contain the model i, $S \cup \{i\}$ plays a role of A in the LOMO; the score associated with the subset S is the difference of measures between F^S and $F^{S \cup \{i\}}$. Then, all scores are aggregated across all possible subsets that the model i does not belong to (Algorithm 2). For example, using the earlier setup of three forecast models, LASOMO considers three subsets, which we denote by $S_1 = \{2\}$, $S_2 = \{3\}$, and $S_3 = \{2, 3\}$, to calculate the importance score of model 1 (excluding all subsets that include model 1). The ensemble forecasts $F^{\{2\}}, F^{\{3\}}$, and $F^{\{2,3\}}$ are then compared to $F^{\{1,2\}}, F^{\{1,3\}}$, and $F^{\{1,2,3\}}$, respectively. The performance differences attributable to model 1's inclusion are aggregated, which results in the importance score for model 1. We note that the subsets (e.g., S_1, S_2, and S_3) may have different weights during the aggregating process.

The **modelimportance** package offers two weighting options for subsets: one assigns equal (uniform) weights to all subsets, and the other assigns weights based on their size, similar to the concept of Shapley values in cooperative game theory, which measure a player's average contribution to all possible coalitions (or, equivalently, over all permutations of players)

(Shapley (1953)). Users can choose one to evaluate the contribution of each model in a manner suited to their preferred framework.

Algorithms 1 and 2 outline the steps to implement LOMO and LASOMO for a single prediction task, respectively.

3.1. Comparison of weighting schemes in LASOMO

The differences in how the two weighting schemes influence the importance scores become more pronounced as the number of models increases. As described in Algorithm 2, the formulas for their subset weights are

$$\frac{1}{2^{n-1}-1} \quad \text{and} \quad \frac{1}{(n-1)\binom{n-1}{k}},$$

where k is the size of each subset and n is the total number of models. The equal scheme treats all subsets equally, so medium-sized subsets have considerable influence in the final result, as there are many such subsets. In contrast, the permutation-based scheme adjusts the weights according to the subset size, giving the greatest weight to both the smallest and largest subsets while assigning small weights to the mid-sized subsets. Moreover, the weights assigned to the mid-sized subsets under the permutation-based approach decrease much faster with n than those under the equal weighting scheme (see Appendix A for details). Consequently, when n is large, middle-sized subsets play a dominant role in determining the importance scores under the equal weighting scheme, whereas extreme-sized subsets primarily drive the scores under the permutation-based weighting approach.

Overall, the difference between the two weighting schemes is likely to arise mainly from the the extreme-sized subsets when n is large. This is because the weights given to the mid-sized subsets become increasingly similar, which are very small values on the order of 10^{-3} even when $n = 8$, while the weights assigned to the smallest and largest subsets remain substantially different (Figure 2).

4. Main functions

In this section, we describe the usage of the function `model_importance()`, where multiple options are available to customize the evaluation framework (Table 3, Table 4).

4.1. model_importance( )

The `model_importance()` function calculates the importance scores of ensemble component models based on their contributions to improving ensemble prediction accuracy for each prediction task. It returns a single data frame of importance scores combined across all tasks. If a model missed predictions for a specific task, an `NA` value will be assigned for that task.

```
> model_importance(forecast_data, oracle_output_data, ensemble_fun,
                   importance_algorithm, subset_wt, min_log_score,
                   ...)
```

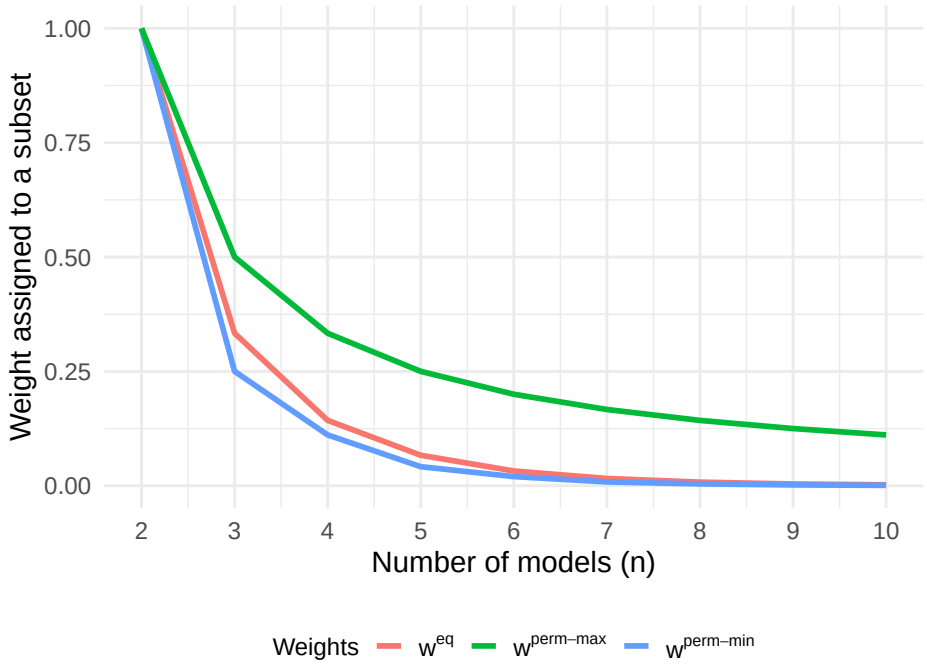

Figure 2: Comparison of weights assigned to a subset. The plot shows the weights assigned to a subset as the number of models n increases from 2 to 10. The red line represents the weights under the equal weighting scheme, while the blue and green lines represent the minimum and maximum weights, respectively, under the permutation-based weighting scheme. The minimum weight occurs when the subset size is around $(n-1)/2$, and the maximum weight occurs when the subset size is $n-1$. As the number of models increases, the weights assigned by the two schemes become increasingly similar for mid-sized subsets whereas substantial differences remain for extreme-sized subsets.

The `forecast_data` is a data frame containing predictions and should be or can be coerced to a `model_out_tbl` format, which is the standard S3 class model output format defined by the hubverse convention. If it fails to be coerced to a `model_out_tbl` format, an error message will be returned from **hubUtils**, which provides the function `as_model_out_tbl()` for this purpose. Users may need to manually transform their data to meet the hubverse standards.

The `oracle_output_data` is a data frame containing the actual observed values of the variables used to specify modeling targets. Details are provided in Section 2.4.

The `ensemble_fun` argument specifies the ensemble method to be used for evaluating model importance, which relies on implementations in the **hubEnsembles** package (Shandross *et al.* (2025)). The currently supported methods are `"simple_ensemble"` and `"linear_pool"`. The `"simple_ensemble"` method returns the average of the predicted values from all component models per prediction task defined by task IDs, `output_type`, and `output_type_id` columns. The default aggregation function for this method is `"mean"`, but it can be customized by specifying additional arguments through `...`, such as `agg_fun="median"`. When `"linear_pool"` is specified, ensemble model outputs are created as a linear pool of component model outputs. This method supports only an `output_type` of `"mean"`, `"quantile"`, or `"pmf"`.

Argument	Description	Possible Values	Default
`forecast_data`	Forecasts	Must be the model output format	N/A
`oracle_output_data`	Ground truth data	Must be the oracle output format	N/A
`ensemble_fun`	Ensemble method	`"simple_ensemble"`, `"linear_pool"`	`"simple_ensemble"`
`importance_algorithm`	Algorithm to calculate importance	`"lomo", "lasomo"`	`"lomo"`
`subset_wt`	Method for assigning weight to subsets when using LASOMO algorithm	`"equal"`, `"perm_based"`	`"equal"`
`min_log_score`	Minimum value to replace for log score	Non-positive numeric	-10
`...`	Optional arguments for `"simple_ensemble"`	Varies	`agg_fun="mean"`

Table 3: Description of the arguments for the `model_importance()` function, including their purpose, possible values, and default settings.

The `importance_algorithm` argument specifies the algorithm for model importance calculation, which can be either `"lomo"` (leave one model out) and `"lasomo"` (leave all subsets of models out). The `subset_wt` argument is employed only for the `"lasomo"` algorithm. This argument has two options: `"equal"` assigns equal weight to all subsets and `"perm_based"` assigns weight averaged over all possible permutations as in the formula of Shapley values (Algorithm 2). The default values of `importance_algorithm` and `subset_wt` are `"lomo"` and `"equal"`, respectively.

The `min_log_score` argument is relevant only for the `output_type` of `"pmf"`, which uses Log Score as a scoring rule. It sets a minimum threshold for log scores to avoid issues with extremely low probabilities assigned to the true outcome, which can lead to undefined or negative infinite log scores. Any probability lower than this threshold will be adjusted to this minimum value before calculating the importance metric based on the log score. The default value is set to -10, following the CDC FluSight thresholding convention (Brooks *et al.* 2018; Reich *et al.* 2019). Users may choose a different value based on their practical needs.

4.2. model_importance_summary( )

The `model_importance_summary()` function summarizes the importance scores produced by `model_importance()` across tasks for each model.

```
> model_importance_summary(importance_scores, by = "model_id",
                           na_action = c("drop", "worst", "average"),
                           fun = mean, ...)
```

The `importance_scores` is a data frame containing model importance scores for individual prediction tasks, as produced by the `model_importance()` function.

Argument	Description	Possible Values	Default
`importance_scores`	Model importance scores produced by `model_importance()`	data frame	N/A
`by`	Grouping variable(s) for summarization	grouping variable(s)	`"model_id"`
`na_action`	Method to handle `NA` values	`"drop"`, `"worst"`, `"average"`	`"drop"`
`fun`	Function to summarize importance scores	summary function	`mean`
`...`	Optional arguments for `"fun"`	depends on `fun`	N/A

Table 4: Description of the arguments for the `model_importance_summary()` function, including their purpose, possible values, and default settings.

The `by` argument specifies the grouping variable(s) for summarization. The default is `"model_id"` to summarize importance scores for each model. Users can also specify other columns present in the `importance_scores` data frame as needed.

The `na_action` argument allows for specifying how to handle `NA` values generated during importance score calculation for each task, occurring when a model did not contribute to the ensemble prediction for a given task by missing its forecast submission. Three options are available: `"worst"`, `"average"`, and `"drop"`. In each specific prediction task, if a model has any missing predictions, the `"worst"` option replaces the `NA` values with the smallest value among other models' importance metrics, while the `"average"` option replaces them with the average of the other models' importance metrics in that task. The `"drop"` option removes the `NA` values, which results in the exclusion of the model from the evaluation for that task.

The `fun` argument specifies a function used to summarize importance scores. `fun = mean` is a default choice, but other summary functions are also applicable (e.g., `fun = median`). Additional arguments can be passed to the summary function `fun` through `...` if needed (e.g., `fun = quantile, probs = 0.25` for a quartile summary).

5. Implementation and availability

This package is implemented in R and distributed via CRAN and GitHhub. We conducted unit tests using the **testthat** package (Wickham 2011) to ensure that all functions work correctly as expected, including those used internally. We also performed continuous integration testing using GitHub Actions to maintain functionality across platforms, including Windows, macOS, and Linux. Integrated GitHub Action, we used **lintr** package to maintain consistent code style, code quality, and detection potential issues. All code changes were systematically reviewed by fellow team members, and this enhanced reliability.

6. Examples

The examples in this section illustrate the use of the `model_importance()` function to evaluate the importance of component models within an ensemble, using various combinations of the arguments described in Section 4. We use some example forecast and target data from the **hubExamples** package, which provides sample datasets for multiple modeling hubs in the hubverse format.

6.1. Example data

The forecast data used here contains forecasts of weekly incident influenza hospitalizations in the US for Massachusetts (FIPS code 25) and Texas (FIPS code 48), generated on November 19, 2022. These forecasts are for two target end dates, November 26, 2022 (horizon 1), and December 10, 2022 (horizon 3), and were produced by three models: 'Flusight-baseline', 'MOBS-GLEAM_FLUH', and 'PSI-DICE'. The output type is `median` and the `output_type_id` column has `NA`s as no further specification is required for this output type. We have modified the example data slightly: some forecasts have been removed to demonstrate the handling of missing values. Therefore, MOBS-GLEAM_FLUH's forecast for Massachusetts on November 26, 2022, and PSI-DICE's forecast for Texas on December 10, 2022, are missing.

```
> forecast_data
```

```
# A tibble: 10 x 9
   model_id          reference_date target          horizon location
   <chr>             <date>         <chr>             <int> <chr>
 1 Flusight-baseline 2022-11-19     wk inc flu hosp       1 25
 2 Flusight-baseline 2022-11-19     wk inc flu hosp       3 25
 3 Flusight-baseline 2022-11-19     wk inc flu hosp       1 48
 4 Flusight-baseline 2022-11-19     wk inc flu hosp       3 48
 5 MOBS-GLEAM_FLUH   2022-11-19     wk inc flu hosp       3 25
 6 MOBS-GLEAM_FLUH   2022-11-19     wk inc flu hosp       1 48
 7 MOBS-GLEAM_FLUH   2022-11-19     wk inc flu hosp       3 48
 8 PSI-DICE          2022-11-19     wk inc flu hosp       1 25
 9 PSI-DICE          2022-11-19     wk inc flu hosp       3 25
10 PSI-DICE          2022-11-19     wk inc flu hosp       1 48
   target_end_date output_type output_type_id value
   <date>          <chr>       <chr>          <dbl>
 1 2022-11-26      median      <NA>              51
 2 2022-12-10      median      <NA>              51
 3 2022-11-26      median      <NA>            1052
 4 2022-12-10      median      <NA>            1052
 5 2022-12-10      median      <NA>              43
 6 2022-11-26      median      <NA>            1072
 7 2022-12-10      median      <NA>             688
 8 2022-11-26      median      <NA>              90
 9 2022-12-10      median      <NA>             159
10 2022-11-26      median      <NA>            1226
```

The corresponding target data contains the observed hospitalization counts for these dates and locations.

```
> target_data
```

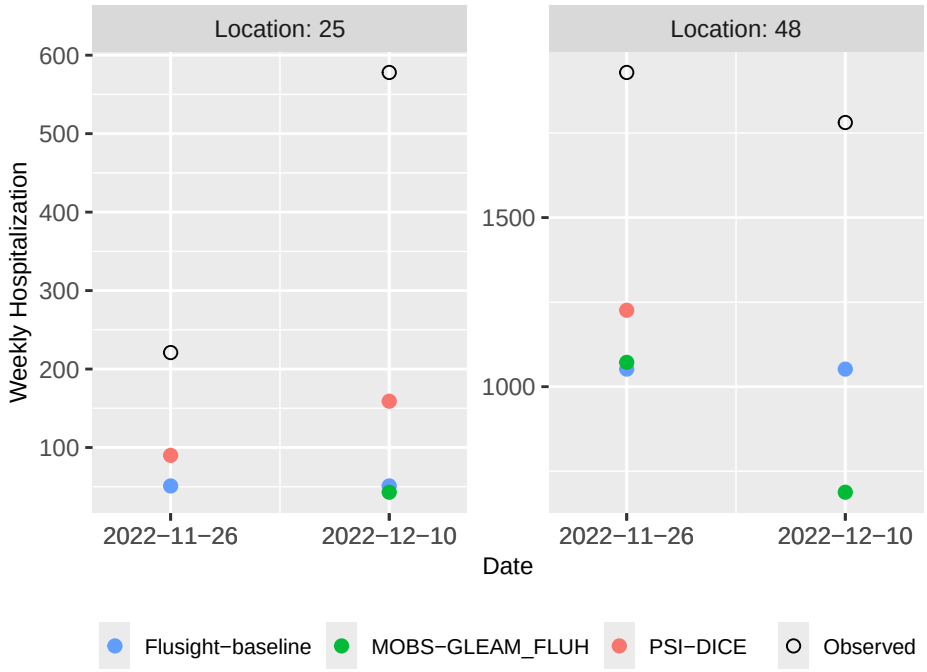

Figure 3: Plot of three point forecasts (median) and the eventually observed values from the `forecast_data` and `target_data` for weekly incident influenza hospitalizations in Massachusetts (FIPS code 25) and Texas (FIPS code 48). Colored dots indicate the forecasts by three models, generated on November 19, 2022. Open black circles indicate the eventually observed values. MOBS-GLEAM_FLUH's forecast for Massachusetts on November 26, 2022, and PSI-DICE's forecast for Texas on December 10, 2022, are missing.

```
# A tibble: 4 x 4
  target_end_date target          location oracle_value
  <date>          <chr>           <chr>           <dbl>
1 2022-11-26      wk inc flu hosp 25                221
2 2022-11-26      wk inc flu hosp 48               1929
3 2022-12-10      wk inc flu hosp 25                578
4 2022-12-10      wk inc flu hosp 48               1781
```

Overall, the forecasts tend to have larger prediction errors for the target end date of December 10, 2022, compared to November 26, 2022, which is expected due to increased uncertainty at longer horizons. Additionally, the forecasts for Massachusetts are relatively more accurate compared to those for Texas, which tend to have higher errors (Figure 3).

6.2. Evaluation using LOMO algorithm

We quantify the contribution of each model within the ensemble using the `model_importance()` function. The following code evaluates the importance of each ensemble member in the simple mean ensemble using the LOMO algorithm.

```
> model_importance(
        forecast_data = forecast_data,
        oracle_output_data = target_data,
        ensemble_fun = "simple_ensemble",
        importance_algorithm = "lomo"
)
```

This call generates both the result and informative messages, summarizing the input data, including the number of dates on which forecasts were produced and the number of models with ids as follows.

```
Forecasts from 2022-11-19 to 2022-11-19 (a total of 1 forecast date(s)).
The available model IDs are:
     Flusight-baseline
     MOBS-GLEAM_FLUH
     PSI-DICE
(a total of 3 models)
```

The function output is a data frame containing model ids and their corresponding importance scores for each prediction task, along with task id columns.

```
            model_id reference_date          target horizon location
1  Flusight-baseline     2022-11-19 wk inc flu hosp       1       25
2    MOBS-GLEAM_FLUH     2022-11-19 wk inc flu hosp       1       25
3           PSI-DICE     2022-11-19 wk inc flu hosp       1       25
4  Flusight-baseline     2022-11-19 wk inc flu hosp       1       48
5    MOBS-GLEAM_FLUH     2022-11-19 wk inc flu hosp       1       48
6           PSI-DICE     2022-11-19 wk inc flu hosp       1       48
7  Flusight-baseline     2022-11-19 wk inc flu hosp       3       25
8    MOBS-GLEAM_FLUH     2022-11-19 wk inc flu hosp       3       25
9           PSI-DICE     2022-11-19 wk inc flu hosp       3       25
10 Flusight-baseline     2022-11-19 wk inc flu hosp       3       48
11   MOBS-GLEAM_FLUH     2022-11-19 wk inc flu hosp       3       48
12          PSI-DICE     2022-11-19 wk inc flu hosp       3       48
   target_end_date output_type importance
1       2022-11-26      median  -19.50000
2       2022-11-26      median         NA
3       2022-11-26      median   19.50000
4       2022-11-26      median  -32.33333
5       2022-11-26      median  -22.33333
6       2022-11-26      median   54.66667
7       2022-12-10      median  -16.66667
8       2022-12-10      median  -20.66667
9       2022-12-10      median   37.33333
10      2022-12-10      median  182.00000
11      2022-12-10      median -182.00000
12      2022-12-10      median         NA
```

For models that missed forecasts for certain tasks, `NA` values were assigned in the importance column for those tasks.

We summarize the importance scores for each model by averaging across all tasks. `NA` values

are removed during the averaging process by setting the `na_action` argument to `"drop"`.

```
> scores_lomo <- model_importance(
        forecast_data = forecast_data,
        oracle_output_data = target_data,
        ensemble_fun = "simple_ensemble",
        importance_algorithm = "lomo"
)

> model_importance_summary(
        importance_scores = scores_lomo,
        by = "model_id",
        na_action = "drop",
        fun = mean
)

# A tibble: 3 x 2
  model_id          importance_score_mean
  <chr>                             <dbl>
1 PSI-DICE                           37.2
2 Flusight-baseline                  28.4
3 MOBS-GLEAM_FLUH                   -75
```

The results show that the model 'PSI-DICE' has the highest importance score, followed by 'Flusight-baseline' and 'MOBS-GLEAM_FLUH'. That is, 'PSI-DICE' contributes the most to improving the ensemble's predictive performance, whereas 'MOBS-GLEAM_FLUH', which has a negative score, detracts from the ensemble's performance. The low importance score of 'MOBS-GLEAM_FLUH' is mainly due to a substantially larger prediction error for Texas on the target end date of December 10, 2022, compared to other models, while its missing forecast for Massachusetts for November 26, 2022, was not factored into the evaluation. This single large error significantly affected its contribution score.

Another approach to handling missing values is to use the `"worst"` option for `na_action`, which replaces missing values with the worst (i.e., minimum) score among the other models for the same task.

```
> model_importance_summary(
        importance_scores = scores_lomo,
        by = "model_id",
        na_action = "worst",
        fun = mean
)

# A tibble: 3 x 2
  model_id          importance_score_mean
  <chr>                             <dbl>
1 Flusight-baseline                  28.4
2 PSI-DICE                          -17.6
3 MOBS-GLEAM_FLUH                   -61.1
```

The results show that the importance scores of 'Flusight-baseline' is unchanged because it has no missing forecast. We observe that the importance score of 'PSI-DICE', which was previously positive, has now decreased to a negative value when compared to the evaluation using the `"drop"` option for `na_action`. Moreover, 'MOBS-GLEAM_FLUH' still ranks the lowest, but the importance score has increased. This change is related to the varying forecast accuracy across different tasks. For the target end date of November 26, 2022, in Massachusetts, most forecasts are relatively accurate. Thus, even if the 'MOBS-GLEAM_FLUH' is assigned the worst value of importance score for its missing forecast, including this value in the averaging is not detrimental to the overall importance metric; rather, it is more beneficial than excluding it. In contrast, for the target end date of December 10, 2022, in Texas, the forecasts have much larger errors across the board, and assigning the worst value of importance score to the missing forecast of 'PSI-DICE' in this task has a detrimental effect on averaging importance scores. This is because the scale of the importance scores is influenced by the magnitude of the prediction errors: for tasks with small errors, the scores remain moderate, while tasks with large errors can yield importance scores of much greater magnitude.

It is also possible to impute the missing scores with intermediate values by assigning the average importance scores of other models in the same task. This strategy may offer a more balanced trade-off by mitigating the influence of the missing data without overly penalizing or overlooking them.

```
> model_importance_summary(
        importance_scores = scores_lomo,
        by = "model_id",
        na_action = "average",
        fun = mean
)
```

```
# A tibble: 3 x 2
  model_id          importance_score_mean
  <chr>                             <dbl>
1 Flusight-baseline                  28.4
2 PSI-DICE                           27.9
3 MOBS-GLEAM_FLUH                   -56.2
```

6.3. Evaluation using LASOMO algorithm

Now we demonstrate the use of the LASOMO algorithm in the evaluation of model importance. As we explored the difference of `na_action` options in the previous LOMO example (Section 6.2), we focus on options for `subset_wt`, which specifies how weights are assigned to subsets of models when calculating importance scores, with `na_action` fixed to `"drop"`.

The following code and corresponding outputs illustrate the evaluation using each weighting scheme.

```
> scores_lasomo_eq <- model_importance(
        forecast_data = forecast_data,
        oracle_output_data = target_data,
        ensemble_fun = "simple_ensemble",
```

```
        importance_algorithm = "lasomo",
        subset_wt = "equal"
)

> model_importance_summary(
        importance_scores = scores_lasomo_eq,
        by = "model_id",
        na_action = "drop",
        fun = mean
)

# A tibble: 3 x 2
  model_id          importance_score_mean
  <chr>                             <dbl>
1 PSI-DICE                           47.4
2 Flusight-baseline                  24.3
3 MOBS-GLEAM_FLUH                   -79.8


> scores_lasomo_perm <- model_importance(
        forecast_data = forecast_data,
        oracle_output_data = target_data,
        ensemble_fun = "simple_ensemble",
        importance_algorithm = "lasomo",
        subset_wt = "perm_based"
)

> model_importance_summary(
        importance_scores = scores_lasomo_perm,
        by = "model_id",
        na_action = "drop",
        fun = mean
)

# A tibble: 3 x 2
  model_id          importance_score_mean
  <chr>                             <dbl>
1 PSI-DICE                           44.8
2 Flusight-baseline                  25.3
3 MOBS-GLEAM_FLUH                   -78.6
```

In this example, there are only three models ($n = 3$), and the weights do not differ significantly between the two weighting schemes. Therefore, the resulting outputs show little difference. However, in general, with a larger number of models, the two weighting schemes may yield quite different importance scores for each model.

In this section, we explored each component model's contribution to the ensemble accuracy with only three models. An extensive application in more complex scenarios with a larger number of models can be found in Kim *et al.* (2024).

7. Summary and discussion

Multi-model ensemble forecasts often provide better accuracy and robustness than single models, and are widely used in decision-making and policy planning across various domains. The contribution of each component model to the accuracy of the ensemble depends on its own unique characteristics. The **modelimportance** package enables the quantification of the value that each component model adds to the ensemble performance in different evaluation contexts.

The primary function of the package is `model_importance()`, which returns a data frame with component models and their importance metrics. Users can choose the various ensemble methods to apply and model importance algorithm between LOMO and LASOMO. Additionally, customizable options are available for handling missing values. These features enable the package to serve as a versatile tool to aid collaborative efforts to construct an effective ensemble model across a wide range of forecasting tasks. We note that unit testing with continuous integration ensures the reliability of all functions and the overall quality of code across multiple platforms.

There is a room to enhance the current version of this package. Although this package supports four different output types ('mean', 'median', 'quantile, and 'pmf'), other output types are widely used in practice. For example, 'sample' output type is commonly used in the US Flu Scenario Modeling Hub (Flu Scenario Modeling Hub 2024). This format includes multiple simulated values (samples) from the forecast distribution. The `output_type_id` is specified for each sample, which typically indexes the samples or indicates their source, depending on the context. The package can be extended to support this output type, which is under consideration for future releases. These extensions would aim to broaden the scope of applications in real-world forecasting tasks.

Acknowledgements

We acknowledge Zhian N. Kamvar for debugging and resolving coding issues while developing the package. We are also grateful to Mattew Cornell for his advice on unit testing, which greatly helped us improve the structure and testing our code with a solid understanding of unit testing practices.

Appendix

A. Weights for subsets in LASOMO

In the LASOMO algorithm, two weighting schemes are available for subsets of models in the calculation of model importance scores: equal weights and permutation-based weights.

Let n be the total number of models and k be the size of a subset that does not include the

model being evaluated. The formulas for the weights under each scheme are as follows:

$$w^{\text{eq}} = \frac{1}{2^{n-1}-1},$$
$$w^{\text{perm}} = \frac{1}{(n-1)\binom{n-1}{k}},$$

where the superscripts "eq" and "perm" denote the equal and permutation-based weighting schemes, respectively. The maximum weight under the permutation-based scheme occurs when $k = n-1$, which is $1/(n-1)$. The minimum weight occurs when the subset size is around $(n-1)/2$ (i.e., $k = \lfloor (n-1)/2 \rfloor$), which is approximately $\frac{\sqrt{\pi(n-1)/2}}{(n-1)2^{n-1}}$ by Stirling's approximation.

Given a fixed mid-sized subset, as n increases, the weight assigned to this subset under the equal weighting scheme decreases at a rate of $O(1/2^n)$, while under the permutation-based scheme, it decreases at a much faster rate of $O(1/(\sqrt{n}\,2^n))$. This indicates that as the number of models grows, that mid-sized subset becomes significantly less influential in determining model importance scores when using the permutation-based weighting scheme compared to the equal weighting scheme.

On the other hand, for subsets of extreme sizes (e.g., $k = 1$ or $k = n-1$), the weights under permutation-based weighting scheme decrease only at $O(1/n)$, much slower under the equal weighting scheme. This implies that in scenarios with a large number of models, the contributions of these extreme-sized subsets play a relatively larger role in the calculation of model importance scores when using permutation-based weights compared to the equal weighting approach.

References

Bosse NI, Gruson H, Cori A, van Leeuwen E, Funk S, Abbott S (2022). "Evaluating forecasts with scoringutils in R." *arXiv preprint arXiv:2205.07090*.

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

Brooks LC, Farrow DC, Hyun S, Tibshirani RJ, Rosenfeld R (2018). "Nonmechanistic forecasts of seasonal influenza with iterative one-week-ahead distributions." *PLOS Computational Biology*, **14**(6), e1006134.

Chen T, He T, Benesty M, Khotilovich V, Tang Y, Cho H, Chen K, Mitchell R, Cano I, Zhou T, Li M, Xie J, Lin M, Geng Y, Li Y, Yuan J (2024). *xgboost: Extreme Gradient Boosting*. R package version 1.7.7.1, URL https://CRAN.R-project.org/package=xgboost.

Consortium of Infectious Disease Modeling Hubs (2024). "The hubverse: open tools for collaborative forecasting." Accessed: 2025-01-14, URL https://hubverse.io.

Cramer EY, Huang Y, Wang Y, Ray EL, Cornell M, Bracher J, Brennen A, Rivadeneira AJC, Gerding A, House K, *et al.* (2022). "The United States COVID-19 forecast hub dataset." *Scientific data*, **9**(1), 462.

Flu Scenario Modeling Hub (2024). "Flu Scenario Modeling Hub." Accessed: 2025-07-25, URL https://fluscenariomodelinghub.org/index.html.

Gneiting T, Raftery AE (2005). "Weather forecasting with ensemble methods." *Science*, **310**(5746), 248–249.

Gneiting T, Raftery AE (2007). "Strictly Proper Scoring Rules, Prediction, and Estimation." *Journal of the American Statistical Association*, **102**(477), 359–378. doi:10.1198/016214506000001437.

Hamner B, Frasco M (2018). *Metrics: Evaluation Metrics for Machine Learning*. R package version 0.1.4, URL https://CRAN.R-project.org/package=Metrics.

Hastie T, Tibshirani R, Friedman J (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

He K, Yang Q, Ji L, Pan J, Zou Y (2023). "Financial Time Series Forecasting with the Deep Learning Ensemble Model." *Mathematics*, **11**(4). doi:10.3390/math11041054.

Jordan A, Krüger F, Lerch S (2019). "Evaluating Probabilistic Forecasts with scoringRules." *Journal of Statistical Software*, **90**(12), 1–37. doi:10.18637/jss.v090.i12.

Jordan A Guerra, Sophie A Murray, D Shaun Bloomfield, Peter T Gallagher (2020). "Ensemble forecasting of major solar flares: methods for combining models." *J. Space Weather Space Clim.*, **10**, 38. doi:10.1051/swsc/2020042.

Kim M, Ray EL, Reich NG (2024). "Beyond forecast leaderboards: Measuring individual model importance based on contribution to ensemble accuracy." URL https://arxiv.org/abs/2412.08916.

Krystalli A, Shandross L (2025). *hubUtils: Core 'hubverse' Utilities*. R package version 0.6.0.9000, commit 71964f7cd6f79e48f3038952fed8f742a7fe6a5c, URL https://github.com/hubverse-org/hubUtils.

Kuhn M (2008). "Building Predictive Models in R Using the caret Package." *Journal of Statistical Software*, **28**(5), 1–26. doi:10.18637/jss.v028.i05. URL https://www.jstatsoft.org/index.php/jss/article/view/v028i05.

Liaw A, Wiener M (2002). "Classification and Regression by randomForest." *R News*, **2**(3), 18–22. URL https://CRAN.R-project.org/doc/Rnews/.

Lutz CS, Huynh MP, Schroeder M, Anyatonwu S, Dahlgren FS, Danyluk G, Fernandez D, Greene SK, Kipshidze N, Liu L, Mgbere O, McHugh LA, Myers JF, Siniscalchi A, Sullivan AD, West N, Johansson MA, Biggerstaff M (2019). "Applying infectious disease forecasting to public health: a path forward using influenza forecasting examples." *BMC Public Health*, **19**(1), 1659. doi:10.1186/s12889-019-7966-8.

O'Hara-Wild M, Hyndman R, Wang E (2024). *fable: Forecasting Models for Tidy Time Series*. R package version 0.4.0, https://github.com/tidyverts/fable, URL https://fable.tidyverts.org.

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

Ray EL, Sweger B, Contamin L (2025). *hubExamples: Example Hub Data*. R package version 0.1.0.9000, commit df75591bff3d68c3ecb1cad4798b1334230203f7, URL https://github.com/hubverse-org/hubExamples.

Reich N, Ray E, Bosse N, Cornell M, Sweger B, Roosa K (2025). *hubEvals: Basic tools for scoring hubverse forecasts*. R package version 0.0.0.9001, commit 0329cb4f94f833c495bc3001cbacf82b103f35d2, URL https://github.com/hubverse-org/hubEvals.

Reich NG, McGowan CJ, Yamana TK, Tushar A, Ray EL, Osthus D, Kandula S, Brooks LC, Crawford-Crudell W, Gibson GC, Moore E, Silva R, Biggerstaff M, Johansson MA, Rosenfeld R, Shaman J (2019). "Accuracy of real-time multi-model ensemble forecasts for seasonal influenza in the U.S." *PLOS Computational Biology*, **15**(11), e1007486. doi:10.1371/journal.pcbi.1007486.

Ridgeway G, Developers G (2024). *gbm: Generalized Boosted Regression Models*. R package version 2.2.2, URL https://CRAN.R-project.org/package=gbm.

Shandross L, Howerton E, Contamin L, Hochheiser H, Krystalli A, of Infectious Disease Modeling Hubs C, Reich NG, Ray EL (2024). "Multi-model ensembles in infectious disease and public health: Methods, interpretation, and implementation in R." *medRxiv*.

Shandross L, Howerton E, Ray EL (2025). *hubEnsembles: Ensemble Methods for Combining Hub Model Outputs*. R package version 1.0.0.9000, commit ca66c89340933124618aad8d100bfddd89637b7d, URL https://github.com/hubverse-org/hubEnsembles.

Shapley LS (1953). "A value for n-person games." *Contribution to the Theory of Games*, **2**.

Sun S, Wang S, Wei Y (2020). "A new ensemble deep learning approach for exchange rates forecasting and trading." *Advanced Engineering Informatics*, **46**, 101160. doi:https://doi.org/10.1016/j.aei.2020.101160.

Viboud C, Sun K, Gaffey R, Ajelli M, Fumanelli L, Merler S, Zhang Q, Chowell G, Simonsen L, Vespignani A (2018). "The RAPIDD ebola forecasting challenge: Synthesis and lessons learnt." *Epidemics*, **22**, 13–21. doi:10.1016/j.epidem.2017.08.002.

Wickham H (2011). "testthat: Get Started with Testing." *The R Journal*, **3**, 5–10. URL https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf.

Yan Y (2024). *MLmetrics: Machine Learning Evaluation Metrics*. R package version 1.1.3, URL https://CRAN.R-project.org/package=MLmetrics.

Affiliation:

Minsu Kim
E-mail: minsu@umass.edu

Li Shandross

Evan L. Ray

Nicholas G. Reich