Prediction of Infectious Disease Epidemics via Feature-Weighted Density Ensembles

Evan L Ray, Krzysztof Sakrejda, Nicholas G Reich

March 2017

Abstract

Introduction

The practice of combining predictions from different models has been used for decades by climatologists and geophysical scientists. These methods have subsequently been adapted and extended by statisticians and computer scientists in a wide array of applications. In recent years, these "ensemble" forecasting approaches are frequently among the top methods used in prediction challenges across a wide range of applications.

Ensembles are a natural choice for noisy, complex, and interdependent systems that evolve over time. In these settings, no one model is likely to be able to capture and predict the full set of complex relationships that drive future observations from a particular system of interest. Instead "specialist" or "component" models can be relied on to capture distinct features or signals from a system which, when combined, represent a nearly complete range of possible outcomes.

Using component models that generate predictive densities for outcomes of interest, we have developed an ensemble method that determines optimal model combinations based on observed data and aspects of the predictive distributions obtained from those component models. We refer to models built using this approach as "feature-weighted" ensembles. Our approach fuses aspects of different ensemble methods: it uses model stacking [Wolpert, 1992], combines predictive densities as in Bayesian Model Averaging [Madigan et al., 1998], and estimates model weights based on features of the system [Sill et al., 2009] using gradient tree boosting [Friedman, 2001].

To illustrate this method, we present time-series forecasts for infectious disease, specifically for influenza in the U.S. For millenia, infectious diseases have been one of the central existential threats to humankind. The international significance of emerging epidemic threats in recent decades, including HIV/AIDS since the 1980s, SARS in 2003, H1N1 in 2009, and Ebola and MERS in 2014-2015, has also increased public awareness of the importance of understanding and being able to predict infectious disease dynamics. With the revolution in data-driven science also occurring in the past few decades, there is now an increased focus on and hope for using statistics to inform public health policy and decision-making in ways that could impact future outbreaks. Some of the largest public health agencies in the world, including the US Centers for Disease Control and Prevention (CDC) have openly endorsed using models to inform decision making, saying: "with models, decision-makers can look to the future with confidence in their ability to respond to outbreaks and public health emergencies."[Centers for Disease Control and Prevention, 2016a]

Development of the methods presented in this manuscript was motivated by the observation that certain prediction models for infectious disease consistently performed better than other models at certain times of year.

We observed in earlier work that early in the U.S. influenza season, simple models of historical incidence often outperformed more standard time-series prediction models such as a seasonal auto-regressive integrated moving average (SARIMA) model[**?**]. However, in the middle of the season, the time-series models showed improved accuracy. We set out to determine whether ensemble methods could use this information about past model performance to improve predictions.

[[TODO: Insert 1-paragraph ensemble methods literature review here.]]

While there are many different methods for combining models, all ensemble models discussed in this paper use an approach called stacking.
In this approach, each of the component models is trained separately in a first stage, and cross-validated measures of performance of those component models are obtained.
Then, in a second stage, a stacking model is trained using the cross-validated performance measures to learn how to optimally combine predictions from the component models.

Using seasonal influenza outbreaks in the US health regions as a case-study, we developed and applied our feature-weighted ensemble model to predict several attributes of the influenza season at each week during the season. Additionally, we compared the feature-weighted ensemble model to simpler ensemble approaches, such as simple averages of component models. By illustrating the utility of this new approach to ensemble forecasting in a setting with complex population dynamics, this work highlights the importance of continued innovation in ensemble methodology.

Methods

This paper presents a novel ensemble framework applied to forecasting specific features of influenza seasons in the U.S. First, we present a description of the influenza data we use in our application and the prediction targets. Next, we discuss the three component models utilized by our ensemble framework. Finally, we turn to the ensemble framework itself, describing our model's specification as well as that of two "baseline" ensemble models.

Data and prediction targets

We obtained publicly available data on seasonal influenza activity in the United States between 1997 and 2016 from the U.S. Centers for Disease Control and Prevention (CDC) (Figure 1). For each of the 10 Health and Human Services regions in the country in addition to the nation as a whole, the CDC calculates and publishes each week a measure called the weighted influenza-like illness (wILI) index. The wILI for a particular region is calculated as the average proportion of doctor visits with influenza-like illness for each state in the region, weighted by state population. During the CDC-defined influenza season (between Morbidity and Mortality Weekly Report [MMWR] week 40 of one year and 20 of the next year), the CDC publishes updated influenza data on a weekly basis. This includes "current" wILI data from two weeks prior to the reporting date, as well as updates to previously reported numbers as new data becomes available. For this analysis, we use only the final reported wILI measures to train and predict from our models.

The CDC defines the influenza season onset as the first of three successive weeks of the season for which wILI is greater than or equal to a threshold that is specific to the region and season. This threshold is the mean

percent of patient visits where the patient had ILI during low incidence weeks for that region in the past three seasons, plus two standard deviations [Centers for Disease Control and Prevention, 2016b]. The CDC provides historical threshold values for each region going back to the 2007/2008 season [{Centers for Disease Control and Prevention}, 2016]. Additionally, we define two other metrics specific to a region-season. The peak incidence is the maximum observed wILI measured in a season. The peak week is the week at which the maximum wILI for the season is observed.

Each predictive distribution was represented by probabilities assigned to bins associated with different possible outcomes. For onset week, the bins are represented by integer values for each possible season week plus a bin for "no onset". For peak week, the bins are represented by integer values for each possible season week. For peak incidence, the bins capture incidence rounded to a single decimal place, with a single bin to capture all incidence over 13.05. Formally, the incidence bins are as follows: [0, 0.05), [0.05, 0.15), ..., [12.95, 13.05), [13.05, ∞). These bins were used in the 2016-2017 influenza prediction contest run by the CDC (cite something).

We measure the accuracy of predictive distributions using the log score. The log score is a proper scoring rule [Gneiting and Raftery, 2007], calculated in our setting as the natural log of the probability assigned to the bin containing the true observation. Proper scoring rules are preferred for measuring the quality of predictive distributions because the expected score is optimized by the true probabilty distribution. We note that for peak week, in some region-seasons the same peak incidence was achieved in multiple weeks (after rounding to one decimal place). In those cases, we calculated the log score as the log of the sum of the probabilities assigned to those weeks; this is consistent with scoring procedures used in the 2016-2017 flu prediction contest run by the CDC (cite something).

Component models

We used three component models to generate probabilistic predictions of the three prediction targets. The first model was a seasonal average model that utilized kernel density estimation (KDE) to estimate a predictive distribution for each target. The second model utilized kernel conditional density estimation (KCDE) and copulas to create a joint predictive distribution for incidence in all remaining weeks of the season, conditional on recent observations of incidence. By calculating appropriate integrals of this joint distribution, we constructed predictive distributions for each of the seasonal targets. The third model used a standard seasonal auto-regressive integrated moving average (SARIMA) implementation. All models were fit independently on data within each region.

Kernel Density Estimation (KDE)

The simplest of the component models uses kernel density estimation [[TODO: cite Silverman]] to estimate a distribution for each target based on observed values of that target in previous seasons within the region of interest. We used Gaussian kernels and the default KDE settings from the `density` function in the `stats` package for R [R Core Team, 2016] to estimate the bandwidth parameter. For the peak incidence target, we fit to log-transformed observations of historical peak incidence. For the onset week prediction target, we estimated the probability of no onset as the proportion of region-seasons in all regions in the training phase where no week in the season met the criteria for being a season onset.

To create an empirical predictive distribution of size N from a KDE fit based on a data vector $\mathbf{y}_{1:K}$ (for example, this might be the vector of peak week values from the K training seasons), we first drew N samples with

replacement from $\mathbf{y}_{1:K}$, yielding a new vector $\tilde{\mathbf{y}}_{1:N}$. We then drew a single psuedo-random deviates from each of N truncated Gaussian distributions centered at $\tilde{\mathbf{y}}_{1:N}$ with the bandwidth estimated by the KDE algorithm. The Gaussians we sampled from were truncated at the lower and upper bounds of possible values for the given prediction target. Finally, we discretized the sampled values to the target-specific bins. These sampled points then make up the empirical predictive distribution from a KDE model. We set the sample size to $N = 10^5$. In theory, this model assigns non-zero probability to every possible outcome; however, in a few cases the empirical predictive distribution resulting from this Monte Carlo sampling approach assigned probability zero to some of the bins.

It is important to note that the predictions from this model do not change as new data are observed over the course of the season.

Kernel Conditional Density Estimation (KCDE)

We used kernel conditional density estimation and copulas to estimate a joint predictive distribution for flu incidence in each future week of the season, and then calculated predictive distributions for each target from that joint distribution. In our implementation, we first used KCDE to obtain separate predictive densities for flu incidence in each future week of the season. Each of these predictive densities gives a conditional distribution for incidence at one future time point given recent observations of incidence and the current week of the season. KCDE can be viewed as a distribution-based analogue of nearest-neighbors regression. We then used a copula to model dependence among those individual predicitive densities, thereby obtaining a joint predicitive density, or a distribution of incidence trajectories in all future weeks.

To predict seasonal quantities (onset, peak timing, and peak incidence), we simulate $N = 10^5$ trajectories of disease incidence from this joint predictive distribution. For each simulated incidence trajectory, we compute the onset week, peak week, and peak incidence. We then aggregate these values to create predictive distributions for each target. This procedure for obtaining predictive distributions for the targets of interest can be formally justified as an appropriate Monte Carlo integral of the joint predictive distribution for disease incidence in future weeks (see [**?**] for details).

Seasonal auto-regressive integrated moving average (SARIMA)

We fit seasonal ARIMA models [Box et al., 2015] to wILI observations transformed to be on the natural log scale. We manually performed first-order seasonal differencing and used the stepwise procedure from the `auto.arima` function in the `forecast` package [Hyndman and Khandakar, 2008] for R to select the specification of the auto-regressive and moving average terms.

Similar to KCDE, forecasts were obtained by sampling $N = 10^5$ trajectories of wILI values over the rest of the season (using the `simulate.Arima` function from the `forecast` package), and predictive distributions of the targets were computed from these sampled trajectories as described above.

Component model training

We used data from 14 seasons (1997/1998 through 2010/2011) to train the models. Data from five seasons (2011/2012 through 2015/2016) were held out when fitting the models and used exclusively in the testing phase.

To avoid overfitting our models, we made predictions for the test phase only once [Hastie et al., 2011].

Estimation of the ensemble models (discussed in the next subsection) requires cross-validated measures of performance of each of the component models in order to accurately gauge their relative performance. For each region, we estimated the parameters of each component model 15 times: 14 fits were obtained excluding one training season at a time, and another fit used all of the training data. For each fit obtained leaving one season out, we generated a set of three predictive distributions, one for each of the prediction targets in the held-out season. We were not able to generate predictions from the SARIMA and KCDE models for some seasons in the training phase because those models used lagged observations from previous seasons that were missing in our data set. The component model fits based on all of the training data were used to generate predictions for the test phase.

Ensemble models

All of the ensemble models we consider in this article work by averaging predictions from the component models to obtain the ensemble prediction. Additionally, these methods are stacked model ensembles because they use leave-one-season-out predictions from the independently estimated component models as inputs to estimate the model weights ([[TODO: cite something – wolpert stacking paper?]]). We begin our discussion of ensemble methods with a general overview, introducing a common set of notation and giving a broad outline of the ensemble models we will use in this article. We then describe our proposed feature-weighted stacking ensemble model specification in more detail.

Overview of ensemble models

A single set of notation can be used to describe all of the ensemble frameworks implemented here. Let $f_m(y|\mathbf{x}^{(m)})$ denote the predictive density from component model m for the value of the scalar random variable Y conditional on observed variables $\mathbf{x}^{(m)}$. For example, Y could represent the peak incidence for a given season and region. In the context of time series predictions, the covariate vector $\mathbf{x}^{(m)}$ may include time-varying covariates such as the week at which the prediction is made or lagged incidence; we suppress that dependence on time in our notation for the sake of simplicity. The superscript $^{(m)}$ reflects the fact that each component model may use a different set of covariates.

The combined predictive density $f(y|\mathbf{x})$ for a particular target can be written as

$$f(y|\mathbf{x}) = \sum_{m=1}^{M} \pi_m(\mathbf{x}) f_m(y|\mathbf{x}^{(m)}). \tag{1}$$

In Equation (1) the π_m are the model weights, which are allowed to vary as a function of observed features in $\mathbf{x}$. We define $\mathbf{x}$ to be a vector of all observed quantities that are used by any of the component models or in calculating the model weights. In order to guarantee that $f(y|\mathbf{x})$ is a probability distribution we require that $\sum_{m=1}^{M} \pi_m(\mathbf{x}) = 1$ for all $\mathbf{x}$.

In the following subsection, we propose a framework for estimating *feature-dependent weights* for a stacked ensemble model. By *feature-dependent* we mean that the weights associated with different component models are driven by observed features or covariates. Although we illustrate the method in the context of time-series predictions, the method could be used in any setting where we wish to combine distribution estimates from

multiple models. Features could include observed data from the system being predicted (such as recent wILI measurements or the time of year at which predictions are being made), observed data from outside the system (for example, recent weather observations), or features of the predictions themselves (e.g. summaries of the predictive distributions from the component models, such as a measure of spread in the distribution, or the time until a predicted peak). Based on exploration of training phase data and *a priori* knowledge of the disease system, we chose three features of the system to illustrate the proposed "feature-weighting" methodology: week of season, component model uncertainty (defined as the minimum number of predictive distribution bins required to cover 90% probability), and wILI measurement at the time of prediction.

We used four distinct methodologies to define weights to use for the stacking models:

1. Equal Weights (**EW**): $\pi_m(\mathbf{x}) = 1/M$. In this scenario, each model contributes the same weight for each target and for all values of $\mathbf{x}$.

2. Constant model weights via degenerate EM (**dEM**): $\pi_m(\mathbf{x}) = c_m$, a constant where $\sum_{m=1}^{M} c_m = 1$ but the constants are not necessarily the same for each model. These weights are estimated using the degenerate estimation-maximization (dEM) algorithm [Lin and Zhu, 2004]. A separate set of weights is estimated for each region and prediction target.

3. Feature-weighted (**FW**): $\pi_m(\mathbf{x})$ depends on features including week of the season and model uncertainty for the KCDE and SARIMA models. A separate set of weighting functions is estimated for each region and prediction target.

4. Feature-weighted with regularization: $\pi_m(\mathbf{x})$ depends on features, but with regularization discouraging the weights from taking extreme values or from varying too quickly as a function of $\mathbf{x}$. A separate set of weighting functions is estimated for each region and prediction target. We fit three variations on this ensemble model, using different sets of features:

    a. (**FW-reg-w**) week of the season;
    b. (**FW-reg-wu**) week of the season and model uncertainty for the KCDE and SARIMA models;
    c. (**FW-reg-wui**) week of the season, model uncertainty for the KCDE and SARIMA models, and incidence (wILI) in the most recent week.

All in all, this leads to 6 ensemble models, summarized in Table 1. The first three of these models (**EW**, **dEM**, and **FW**) can be viewed as variations on **FW-reg-wu** if we vary the amount and type of regularization imposed on the **FW-reg-wu** model. Thus, comparisons among these four models will allow us to explore the benefits of allowing the model weights to depend on covariates while imposing an appropriate amount of rigidity on the model weight functions $\pi_m(\mathbf{x})$. We will discuss the regularization strategies used in **FW-reg-wu** further in the next subsection. Meanwhile, comparisons among the **FW-reg-w**, **FW-reg-wu**, and **FW-reg-wui** models will allow us to explore the relative contributions to predictive performance that can be achieved by allowing the model weights to depend on different features.

Each of the six ensemble models, along with the three component models, are used to generate predictions in every season-week of each of the five testing seasons, assuming perfect reporting. These predictions are then used to evaluate the prospective predictive performance of each of the ensemble methods. In total, we evaluate 9 models in 11 regions over 5 years and 3 targets of interest.

Feature-weighted stacking framework

In this section we introduce the particular specification of the parameter weight functions $\pi_m(\mathbf{x})$ that we use and discuss estimation.

In order to ensure that the the π_m are non-negative and sum to 1 for all values of $\mathbf{x}$, we parameterize them in terms of the softmax transformation of real-valued latent functions ρ_m:

$$\pi_m(\mathbf{x}) = \frac{\exp\{\rho_m(\mathbf{x})\}}{\sum_{m'=1}^{M} \exp\{\rho_{m'}(\mathbf{x})\}}. \tag{2}$$

For a pair of models $l, m \in \{1, ..., M\}$, $\rho_l(\mathbf{x}) > \rho_m(\mathbf{x})$ indicates that model l has more weight than model m for predictions at the given value of $\mathbf{x}$.

The functions $\rho_m(\mathbf{x})$ could be parameterized and estimated using many different techniques, such as a linear specification in the features, splines, or so on. We have chosen to estimate the functions $\rho_m(\mathbf{x})$ using gradient tree boosting.

Gradient tree boosting uses a forward stagewise additive modeling algorithm to iteratively and incrementally construct a series of regression trees that, when added together, create a function designed to minimize a given loss function. In our application, the algorithm builds up the $\rho_m(\mathbf{x})$ that minimize the negative cross-validated log-score of the stacked prediction $f(y|\mathbf{X})$. We have used the `xgb.train` function in the `xgboost` package ([[TODO: cite]]) for `R` to perform this estimation.

Specifically, we define a single tree as

$$T(\mathbf{x}; \boldsymbol{\theta}) = \sum_{j=1}^{J} \gamma_j I_{R_j}(\mathbf{x}), \tag{3}$$

where the R_j are a set of disjoint regions that comprise a partition of the space $\mathcal{X}$ of feature values $\mathbf{x}$, and I is the indicator function taking the value 1 if $\mathbf{x} \in R_j$ and 0 otherwise. The parameters $\boldsymbol{\theta} = (\boldsymbol{\psi}, \boldsymbol{\gamma})$ for the tree are the split points $\boldsymbol{\psi}$ partitioning $\mathcal{X}$ into the regions R_j and the regression constants $\boldsymbol{\gamma}$ associated with each region. The function $\rho_m(\mathbf{x})$ is obtained as the sum of B trees:

$$\rho_m(\mathbf{x}; \Theta_m) = \sum_{b=1}^{B} T(\mathbf{x}; \boldsymbol{\theta}_{m,b}). \tag{4}$$

In each iteration b of the boosting process, we estimate M new regression trees, one for each component model. These trees are estimated so as to minimize a local approximation to the loss function around the weight functions that were obtained after the previous boosting iteration.

Gradient tree boosting is appealing as a method for estimating the functions ρ_m because it offers a great deal of flexibility in how the weights can vary as a function of the features $\mathbf{x}$. On the other hand, this flexibility can lead to overfitting the training data. In order to limit the chances of overfitting, we have explored the use of four regularization parameters:

1. The number of boosting iterations B. As B increases, more extreme weights (close to 0 or 1) and more rapid changes in the weights as $\mathbf{x}$ varies are possible.

2. An L_1 penalty on the number of tree leaves, J. A large penalty encourages the regression trees to have fewer leaves, so that there is less flexibility for the model weights to vary as a function of $\mathbf{x}$.

3. The maximum depth of the regression trees; i.e., how many times the same region may be successively split. Similarly to the L_1 penalty on the number of tree leaves, this controls how much flexibility there is for the model weights to vary in $\mathbf{x}$. Additionally, the maximum depth controls the number of interactions. For instance, a maximum depth of 2 indicates that there can be at most two-way interactions between different features in setting model weights.

4. An L_1 penalty on the regression constants γ_j. A large penalty encourages these constants to be small, so that the overall model weights change less in each boosting iteration.

We selected values for these regularization parameters using a grid search optimizing leave-one-season-out cross-validated model performance.

[[TODO: Discuss how varying some of these regularization parameters can give the simpler EW and dEM models?? Or ignore? Or defer to supplement?]]

Software and code

We used R version 3.2.4 (2016-03-10) for all analyses.[R Core Team, 2016] All data and code used for this analysis is freely available in an R package online at https://github.com/reichlab/adaptively-weighted-ensemble and may be installed in R directly. Predictions generated in real-time with early development versions of this model during the 2016/2017 influenza season may be viewed at https://reichlab.github.io/flusight/. To maximize reproducibility of our work, we have set seeds prior to running code that relies on stochastic simulations using the `rstream` package.[Leydold, 2015]
Additionally, the manuscript itself was dynamically generated using RMarkdown.

Results

To evaluate overall model performance, we computed the average log-score for each model across all regions and/or test seasons. Additionally for the peak wILI targets, we computed the average log-score for each model in the test seasons separately before and after the peak week. Similarly for the onset week target, we computed the average log-score for each model in the test seasons before and after the onset week.

Figure: 9 panel grid (3 component models x 3 metrics) showing a solid line for each region that represents the average log score across all seasons

```
## Joining, by = "mdl"
```

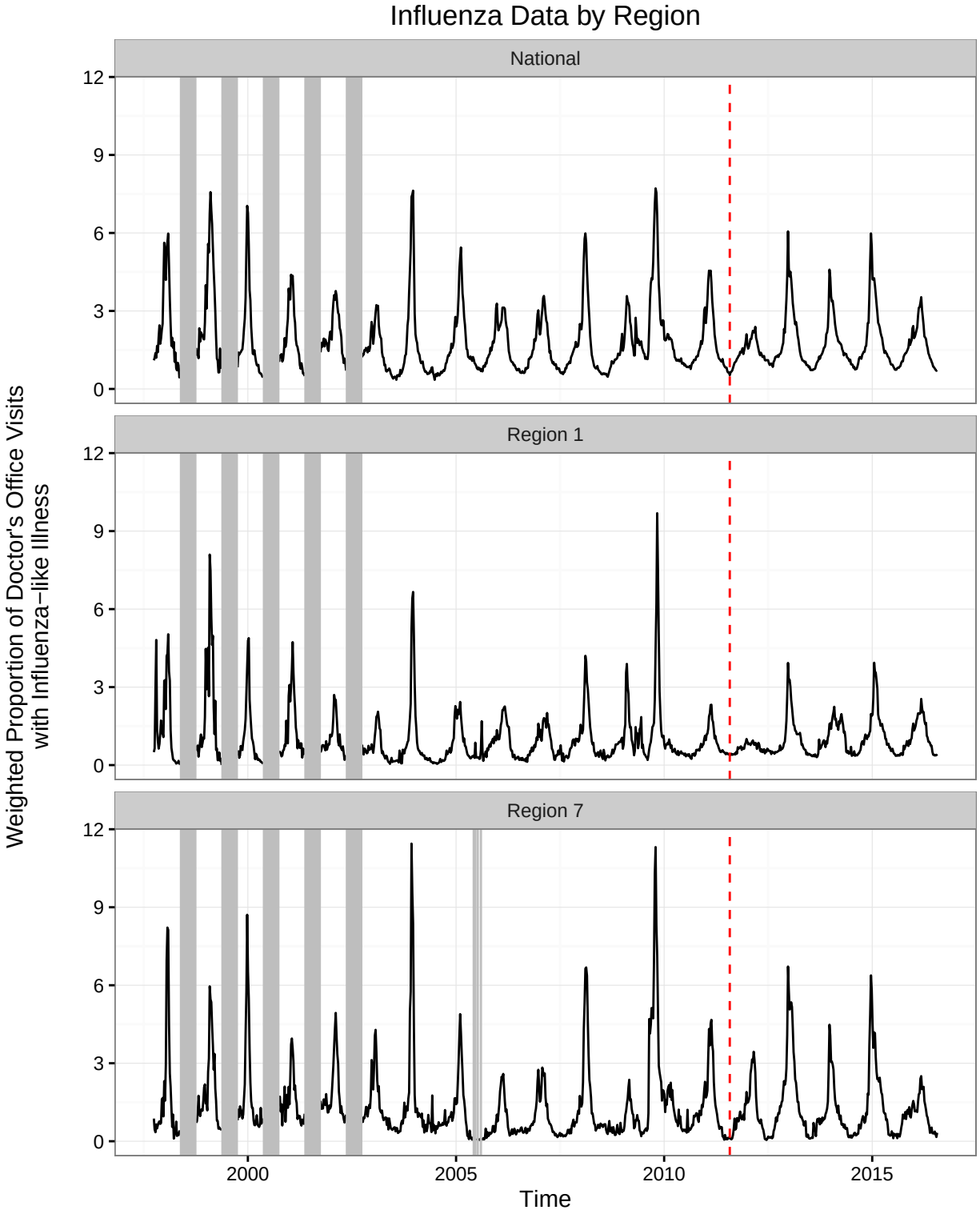
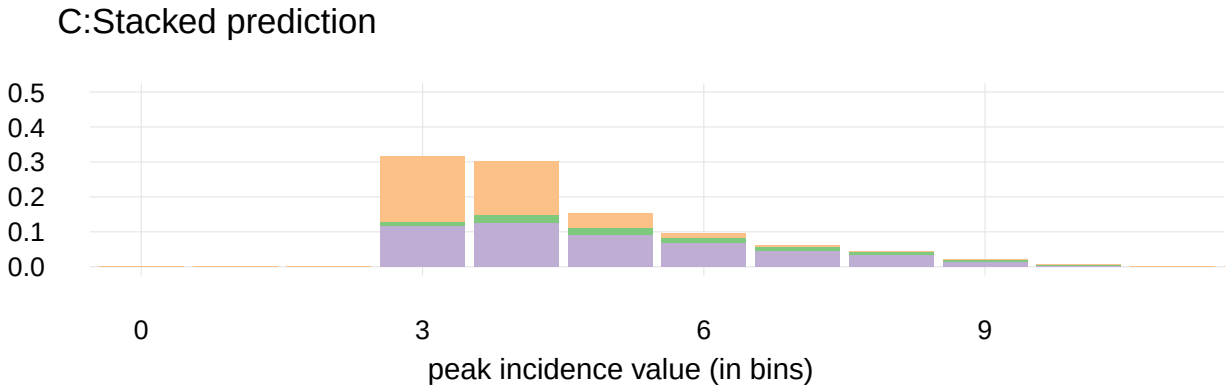


Figure 1: Plot of influenza data. The full data include observations aggregated to the national level and for 10 smaller regions. Here we plot only the data at the national level and in two of the smaller regions; data for the other regions are qualitatively similar. Missing data are indicated with vertical grey lines. The vertical red dashed lines indicate the cutoff time between the training and testing phases; 5 seasons of data were held out for testing.

```
## Warning in left_join_impl(x, y, by$x, by$y, suffix$x, suffix$y): joining
## factor and character vector, coercing into character vector
```



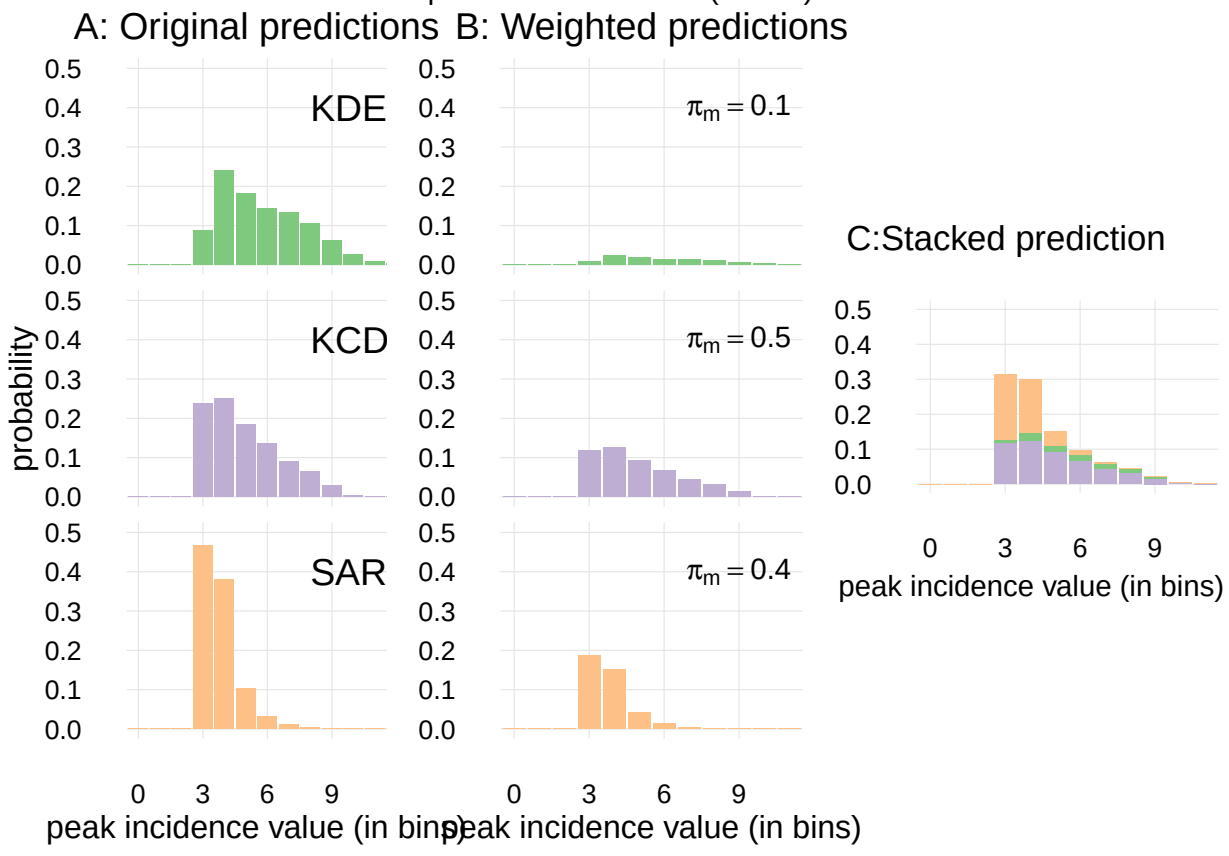


Figure: Example of log-scores and estimated weights from one region by season-week (x) for each model (color). Panel 1 has log-scores (y); Panel 2 has estimated weights (y) using degenerate EM, feature-weighted, and feature-weighted + smoothed

Figure: test phase summary: 3 row-facets, one for each target, each model (color?) year (x) is a point with log-score (y)

```
## Warning: Removed 32 rows containing non-finite values (stat_boxplot).
```

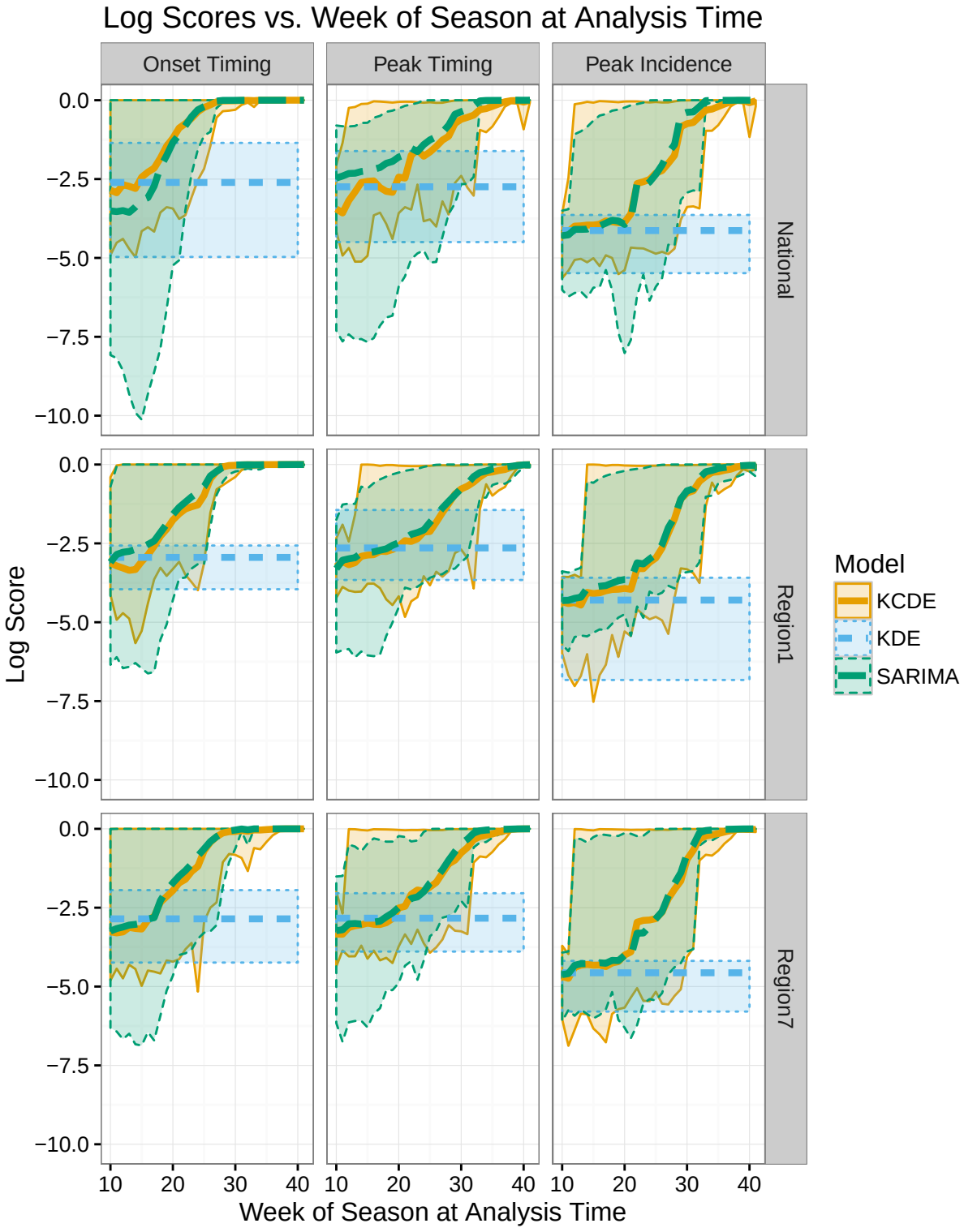


Figure 2: Mean, minimum, and maximum log scores achieved by each component model in each week of the season, summarizing across all seasons in both the train and test phases when all three component models produced predictions.

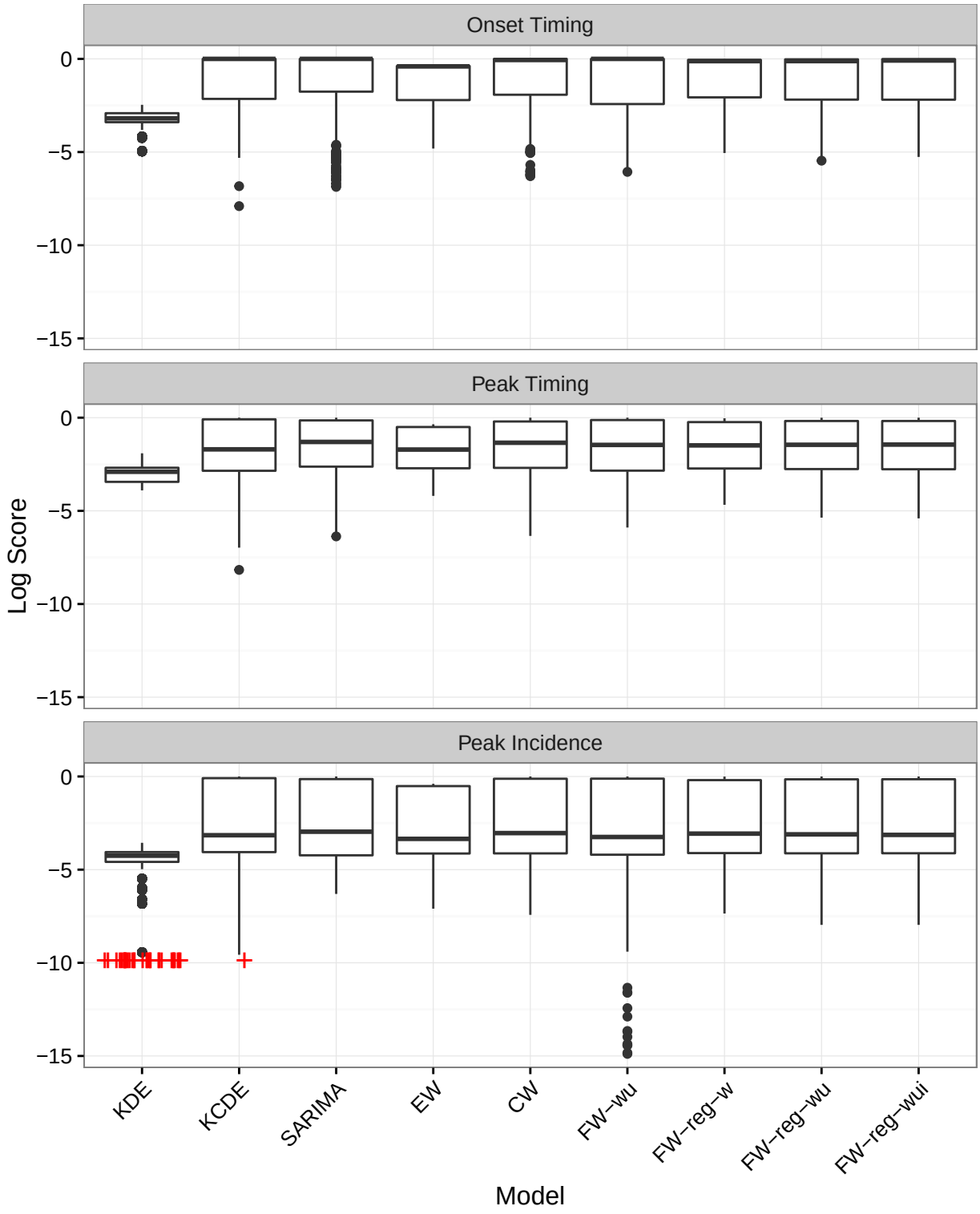


Figure 3: Test phase log scores summary v1. Log scores across all seasons/season weeks represented in box plots. Log scores of -Infinity are represented with a plus sign at -10.

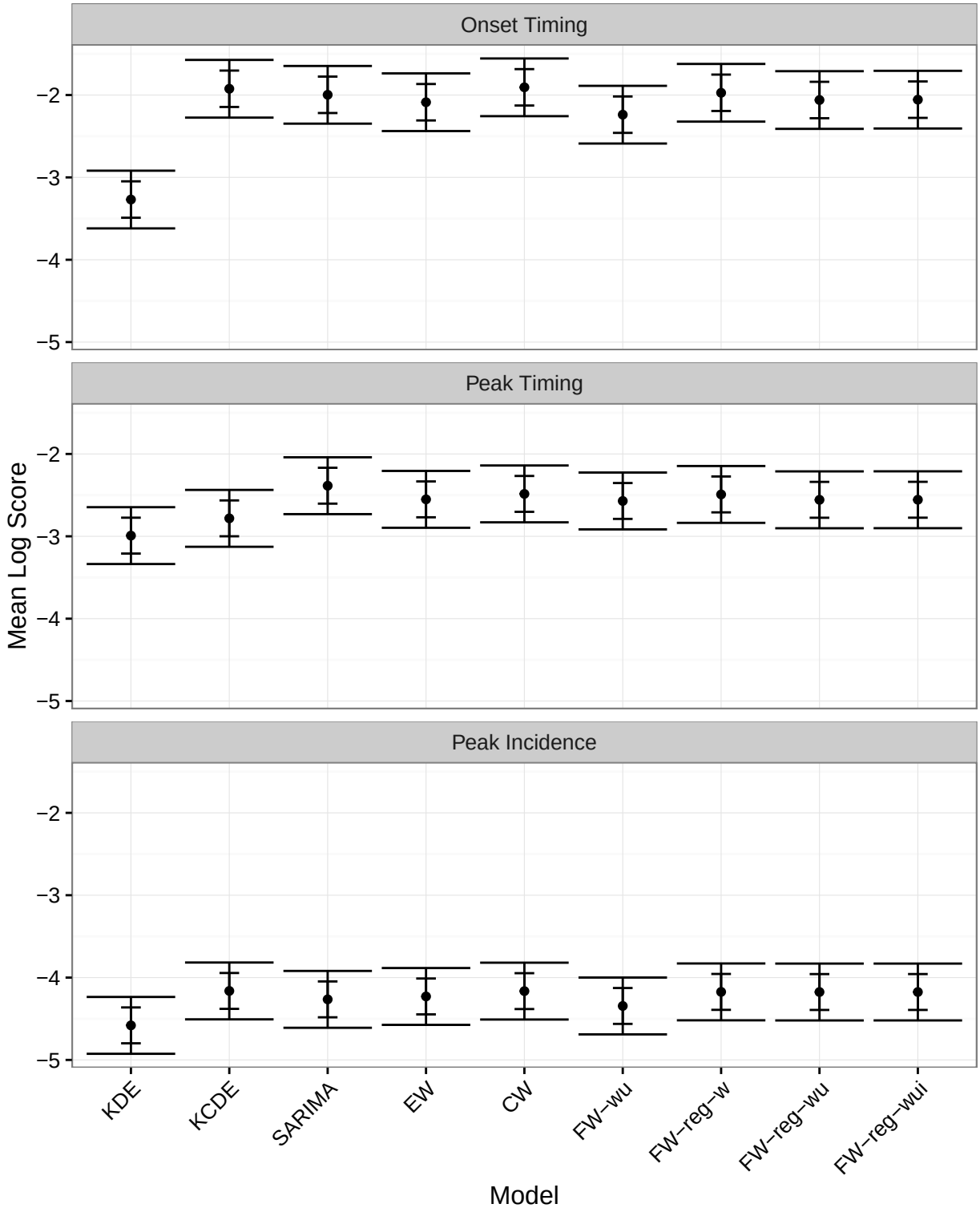


Figure 4: Test phase log scores summary v2. Point estimates and confidence intervals for mean log score for each model in weeks before the target (onset or peak) occurred. Estimates are obtained from a mixed effects model with a separate fixed effect mean for the interaction of model and prediction target; random effects for each combination of region, season, model, and prediction target; and lag 1 autocorrelation nested within each combination of region, season, model, and prediction target. The wider confidence interval bounds are simultaneous confidence intervals with an approximate familywise 95% coverage rate for all intervals. The inner confidence intervals are calculated separately, with approximate coverage rates of 95%. Log scores of -Infinity were truncated at -10 before fitting this model. This model could be fit separately to obtain estimates for mean log scores in weeks on or after the target occurred.

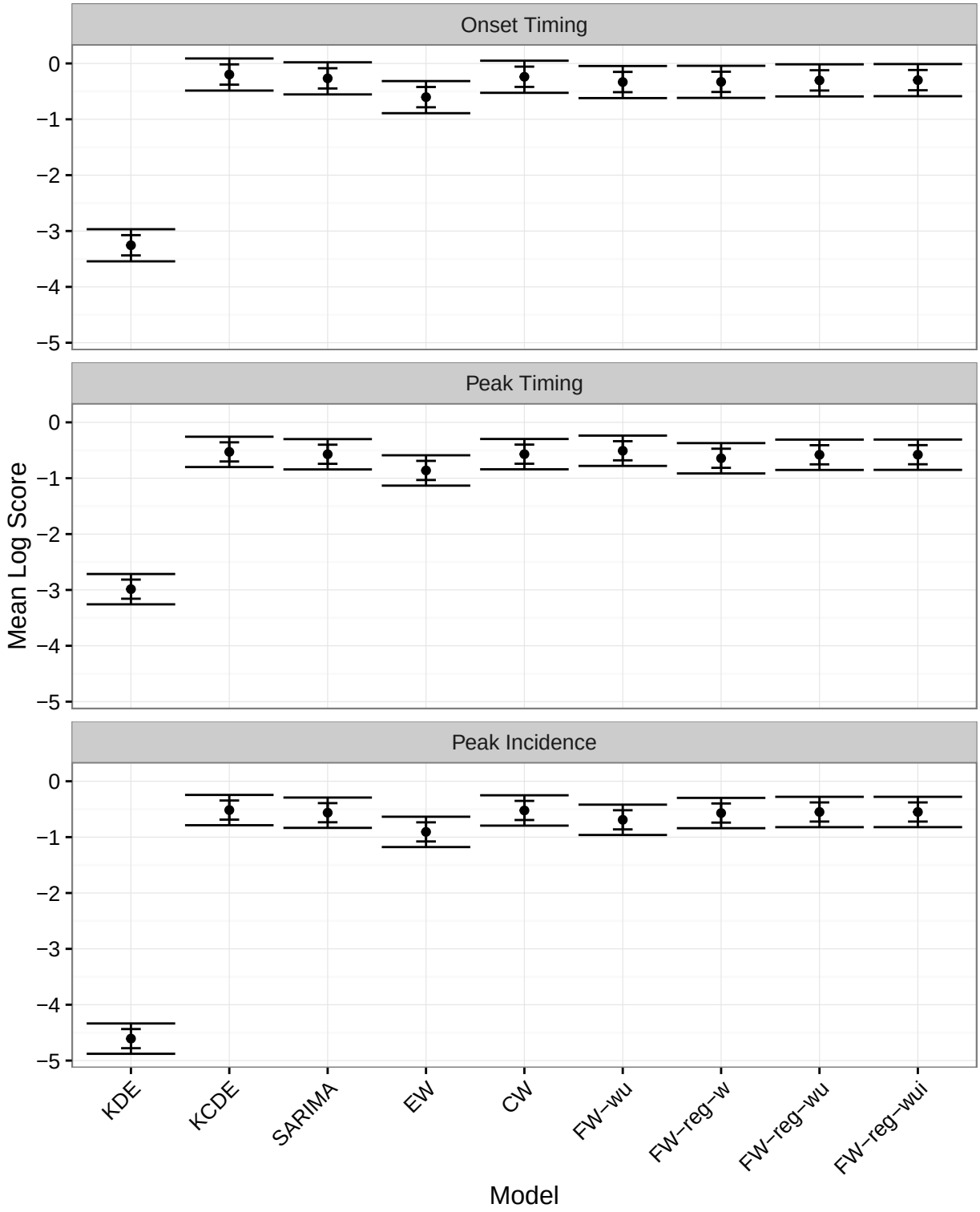


Figure 5: Test phase log scores summary v2. Point estimates and confidence intervals for mean log score for each model in weeks on or after the target (onset or peak) occurred. Estimates are obtained from a mixed effects model with a separate fixed effect mean for the interaction of model and prediction target; random effects for each combination of region, season, model, and prediction target; and lag 1 autocorrelation nested within each combination of region, season, model, and prediction target. The wider confidence interval bounds are simultaneous confidence intervals with an approximate familywise 95% coverage rate for all intervals. The inner confidence intervals are calculated separately, with approximate coverage rates of 95%. Log scores of -Infinity were truncated at -10 before fitting this model.

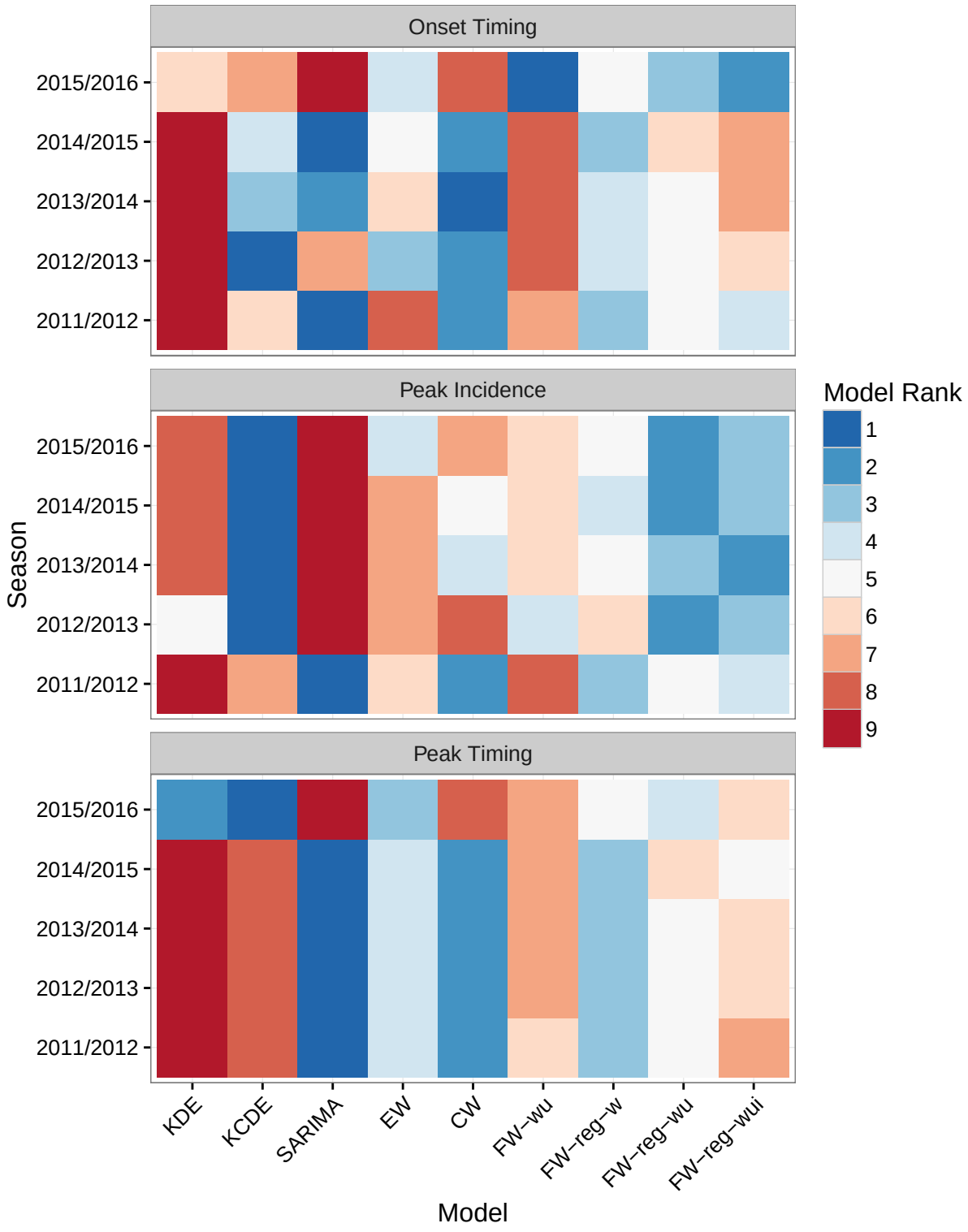


Figure 6: Model performance ranked by mean log score within each of the five test seasons for predictions made before the target (season onset or peak) occurred.

Figure: test phase summary: 3 row-facets, one for each target, each model (color?) year (x) is a point with MAE (y)

Conclusion

Achievements

- developed ensemble framework that makes [[better]] predictions on average than component models
- ensemble method uses novel method to estimate feature-dependent weights
- predictions disseminated and updated weekly

Strengths/novelty

- feature-weighting works regardless of discrete/continuous covariate
- operates on predictive distributions, not just point-estimates
- General framework (not just time-series or infectious disease)
- framework could be used to answer epidemiologically relevant questions: when do certain models contain information that substantially improve predictions, e.g. adding weather, mechanistic models, strain specific models, etc. . .

Limitations

- illustration only includes simple models and simple features

References

George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Centers for Disease Control and Prevention. Staying Ahead of the Curve: Modeling and Public Health Decision-Making, 2016a. URL http://www.cdc.gov/cdcgrandrounds/archives/2016/january2016.htm.

Centers for Disease Control and Prevention. Overview of Influenza Surveillance in the United States, 2016b. URL https://www.cdc.gov/flu/weekly/overview.htm.

{Centers for Disease Control and Prevention}. Regional baseline values for influenza-like illness, 2016. URL https://github.com/cdcepi/FluSight-forecasts/blob/master/wILI{_}Baseline.csv.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. URL http://gateway.webofknowledge.com/gateway/Gateway.cgi?GWVersion=2{&}SrcAuth=mekentosj{&}SrcApp=

Papers{&}DestLinkType=FullRecord{&}DestApp=WOS{&}KeyUT=000244361000032papers2://publication/doi/10.1198/016214506000001437.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, second edition, 2011. URL http://www.amazon.com/Elements-Statistical-Learning-Prediction-Statistics/dp/0387848576/ref=sr{_}1{_}14?ie=UTF8{&}qid=1429565346{&}sr=8-14{&}keywords=machine+learning.

Rob J. Hyndman and Yeasmin Khandakar. Automatic time series forecasting: The forecast package for R. *Journal Of Statistical Software*, 27(3):C3–C3, 2008. ISSN 10411135. doi: 10.18637/jss.v027.i03. URL http://www.robjhyndman.com/papers/forecastpackage.pdf.

Josef Leydold. *rstream: Streams of Random Numbers*, 2015. URL https://CRAN.R-project.org/package=rstream. R package version 1.3.4.

Xiaodong Lin and Yu Zhu. Degenerate Expectation-Maximization Algorithm for Local Dimension Reduction. In *Classification, Clustering, and Data Mining Applications*, pages 259–268. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. doi: 10.1007/978-3-642-17103-1_25. URL http://link.springer.com/10.1007/978-3-642-17103-1{_}25.

David Madigan, A Raftery, Volinsky, and J Hoeting. Bayesian Model Averaging. *Proceedings of the AAAI Workshop on Integrating Multiple Learned Models, Portland, OR.*, (9814), 1998.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016. URL https://www.R-project.org/.

Joseph Sill, Gabor Takacs, Lester Mackey, and David Lin. Feature-Weighted Linear Stacking. nov 2009. URL http://arxiv.org/abs/0911.0460.

David H Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992. URL http://pubget.com/site/paper/2a043750-5667-424d-94f1-2e4e39fccbcc?institution=papers2://publication/doi/10.1016/S0893-6080(05)80023-1.

	Component Model Weights Vary with...					
Model	Region	Prediction Target	Week of Season	SARIMA Uncertainty	KCDE Uncertainty	Current wILI
EW						
dEM	X	X				
FW	X	X	X	X	X	
FW-reg-w	X	X	X			
FW-reg-wu	X	X	X	X	X	
FW-reg-wui	X	X	X	X	X	X

Table 1: Summary of ensemble methods and what the model weights depend on.