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 used among the top methods in prediction challenges.

Ensembles are a natural choice for noisy and complex interdependent systems that evolve over time. In this setting, no one model is likely to be able to predict the full dynamics of a complex system. Instead "specialist" or "component" models could be relied on to capture distinct features or signals from a system which, when combined, represent a complete range of possible outcomes.

Using component methods that generate predictive densities for outcomes of interest, we have developed a feature-weighted density ensemble method that estimates model weights as functions of observed data. 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 [Hastie et al., 2011].

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. Even in today's modern and scientific world, the spectre of a global pandemic appears to be a real possibility. 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." [ADD CITATION: http://www.cdc.gov/cdcgrandrounds/archives/2016/january2016.htm]

The observation that motivated the development of the method presented in this manuscript was seeing how certain prediction models for infectious disease consistently performed better than other models at certain times of year. We set out to determine whether past model performance could improve predictions using feature-weighted

model stacking. We observed 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.

Using seasonal influenza outbreaks in the US health regions as a case-study, we developed and applied our feature-weighted ensemble model to predicting several features of the influenza season at each week during the season.

Methods

Data

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) (). 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 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, 2016] 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 week is the week at which the maximum wILI for the season is observed. The peak incidence is the maximum observed wILI measured in a season.

Component models

We used three component models to generate probabilistic predictions of these three quantities of interest. The first model was a seasonal average model that utilized kernel density estimation to estimate a predictive distribution for each target. The second model utilized kernel conditional density estimation to create predicted trajectories of incidence. By aggregating across all trajectories, we constructed predictive distributions for each target. The third model used a standard seasonal auto-regressive integrated moving average (SARIMA) implementation and model selection approach to fit a model to the data. All models were fit independently on data within a particular region.

Kernel Density Estimation (KDE)

The simplest of the component models uses kernel density estimation (KDE) to estimate a predictive distribution of future values based on observed data from previous seasons within the region of interest. Let the vector of observations for one of the targets be $\mathbf{y}_{1:K}$, so for example, this might represent the vector of the peak wILI values from the K training seasons. We used Gaussian kernels and the default KDE settings from the `density` function in the `stats` package for R [R Core Team, 2016]. To create an empirical predictive distribution of size N from a KDE fit based on a data vector of length K ($\mathbf{y}_{1:K}$), 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 Gaussian distributions centered at $\tilde{\mathbf{y}}_{1:N}$ with the bandwidth estimated by the KDE algorithm. These sampled points then make up the empirical predictive distribution from a KDE model. For the onset week outcome, we added the probability of no onset occuring within a season as [[TODO: XXXX]] and then standardized this along with the sampled probabilities so they summed to 1.

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. This is a strictly seasonal model, only ever predicting a distribution that relates to the average observations in past seasons.

Kernel Conditional Density Estimation (KCDE)

We use a method called kernel conditional density estimation (KCDE) to estimate predictive distributions for each target. Using a kernel-based approach to nearest-neighbors regression, KCDE fits separate predictive densities for each future week in the season. Our implementation for this paper conditioned on recent past incidence and week of season, only using information from the time series itself. In order to predict seasonal quantities (onset, peak timing, and peak incidence), we use a copula to model dependence among those individual predicitive densities for each week, thereby obtaining a joint predicitive density, or a distribution of incidence trajectories in all future weeks. For each predicted trajectory, the peak week, peak height, and onset week are computed and these are aggregated to create predictive distributions for each target respectively (see [**?**] for details).

[[TODO: (Optional) add 2-3 sentences with a few more details on methodology, etc... cross-validation, bandwidth selection, periodic kernel, data on log-scale?, etc...]]

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 used the stepwise procedure from the `auto.arima` function in the `forecast` package for R to select the order of first- and seasonal-differencing as well as the specification of the order of auto-regressive and moving average terms for the SARIMA model.[Hyndman and Khandakar, 2008] Additionally, data were Box-Cox transformed prior to modeling.

[[TODO: Specifications for forecast algorithm - what method used?]] Similar to KCDE, forecasts were obtained as trajectories of wILI values over the rest of the season and predictive distributions of the targets were computed as described above.

Model training and cross-validation

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. Care was taken to ensure that our test predictions were made only once, to avoid overfitting our models [Hastie et al., 2011].

For each model-region pair, we fitted the model 14 times, each time excluding one training season during the fitting. Then for each season-week within the left-out season, we generated a set of three predictive distributions, one for each of the prediction targets. In total, this generated nearly 45,000 predictive distributions across the entire training phase: 3 targets x 3 models x 11 regions x 14 training seasons x 32 weeks per season. (Several of the training seasons had 33 weeks per season because they contained an MMWR week 53.)

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 are intervals of length 0.1 from [0, 0.1), [0.1, 0.2), ..., [12.9, 13.0), with a final bin that captures all high incidence values, [13.0, ∞]. [[TODO: Is this capturing all of the subtlety of rounding in bins?]]

Using the eventually observed value for each of the three outcomes, we then calculated the log-score achieved by each of predictive distributions across the entire training period. 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.

Ensemble model overview

By definition, ensemble models combine a set of models into one single model. While there are many different methods for combining models, all ensemble models discussed in this paper use a model averaging methodology called "stacking", whereby predictions from different models are averaged to obtain the ensemble prediction.[Hastie et al., 2011] In particular, we considered only models that create probabilistic predictive distributions as described above. We implemented several simple stacking procedures to use as reference ensemble methods for our proposed stacking methodology. This ensured that the new methodology could be compared with less complex methods as a baseline.

We propose a novel framework for estimating *feature-dependent weights* for a stacked ensemble model. By *feature-dependent* we mean that the optimal 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 also be used for any model using stacking to generate predictive distributions. 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 distribution itself, such as a measure of spread in the distribution, or the time since 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, model confidence (defined as the minimum number of predictive distribution bins required to cover 90% probability), and wILI measurement at the time of prediction.

All of the ensemble frameworks implemented here can be described using a single set of notation. Let $f_m(y|\mathbf{X})$ denote the predictive density from model m for the value of the scalar random variable Y conditional on observed variables $\mathbf{X}$.

For example, Y could represent the peak incidence for a given season. In the context of repeated predictions from a time-series, the covariate vector $\mathbf{X}$ may include time-dependent covariates, e.g. the week at which the prediction is made. However to introduce the method more generally, we can think of $\mathbf{X}$ as being any set of covariates that may influence which model may more accurately predict the true outcome.

The combined predictive density for a particular target in a given region can be written as

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

In Equation (1) the π_m are the model weights. These π_m could either be taken as *a priori* fixed values (e.g. $\pi_m = 1/M$, for all m) or it could take a more complicated functional form such as

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

where $\rho_m(\mathbf{X})$ is a function that depends on observed data and could be estimated using various techniques. In either setting, the weights must be non-negative and sum to 1 across m.

Using component model predictions from the training phase, we used four distinct methodologies to create or define weights to use for the stacking models.

1. Equal model weights, i.e. $\pi_m = 1/M$. In this scenario, each model contributes the same weight for each target and for all values of $\mathbf{X}$.

2. Constant model weights, i.e. $\pi_m(\mathbf{X}) = c_m$, a constant, where $\sum c_m = 1$ but the constants are not necessarily the same for each model. These weights are estimated using the degenerate estimation-maximization (EM) algorithm.[Lin and Zhu, 2004]

3. Feature-weighted, i.e. $\rho_m(\mathbf{X})$ are estimated using gradient boosting with features including week of the season and model confidence. [[TODO: do we include a season-week only model?]]

4. Feature-weighted and smoothed, i.e. $\rho_m(\mathbf{X})$ as above but with regularization tuning parameters chosen for the gradient boosting that penalize large fluctuations in the estimated $\rho_m(\mathbf{X})$. In these models, we include the same features as in 3 above along with the most recent wILI value. Details on both feature-weighted frameworks are described below.

[[TODO: create a table defining models and highlighting which covariates are used]]

All in all, this leads to [[TODO: 6 or 7]] ensemble models fitted to the training phase data. Each of the 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 [[TODO: total_ensemble_models + total_component_models]] models in 11 regions over 5 years and 3 targets of interest.

Feature-weighted stacking framework

This method was motivated by our observation in earlier work that there are nonlinearities in the relative performance of different models as a function of the time of year at which predictions are made. The framework described above in Equation (**??**) could be used to capture these nonlinearities. Specifically, we 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 log-score of the stacked prediction $f(y|\mathbf{X})$.

Parameters of the algorithm: depth of trees (number of splits within each tree?), number of boosting iterations (is this N, below? like a total number of trees), "learning rate".

Specifically, we define a single tree as

$$T(\mathbf{X};\Theta) = \sum_{j=1}^{J} \gamma_j I(\mathbf{X} \in R_j) \tag{3}$$

where the R_j are a disjoint set of regions that comprise a partition of the space of all predictor variables. Additionally, the model takes $\Theta = \{R_j, \gamma_j\}_{1:J}$ as parameters with J as a meta-parameter. Then the sum of the N trees represents the function

$$\rho_m(\mathbf{X}) = \sum_{n=1}^{N} T(\mathbf{X};\Theta_n). \tag{4}$$

[[TODO: check formulation above, including mention of whether estimation for each model m is done separately or jointly]]

Additionally, we have introduced regularization into our estimation of the ρ_m functions. For now, weight regularization is done through three parameters; optimal values of those parameters are selected through cross-validation and a single best weighting scheme is selected.
The parameters are the depth of the regression trees used in the gradient tree boosting process, the number of boosting iterations, and the learning rate.

We ensure that the the π_m are non-negative and sum to 1 by parameterizing the π_m in terms of the softmax transformation of real-valued functions ρ_m in Equation (2).
We avoid identifiability problems by fixing $\rho_M(\mathbf{X}) = 0$ for all $\mathbf{X}$.
Therefore for models $m \in \{1, ..., M-1\}$, $\rho_m(\mathbf{X}) > 0$ indicates that model m has more weight than the baseline model for predictions at the given value of $\mathbf{X}$.

Model testing

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.

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

Figure: show the flu data

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

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).
```

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

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

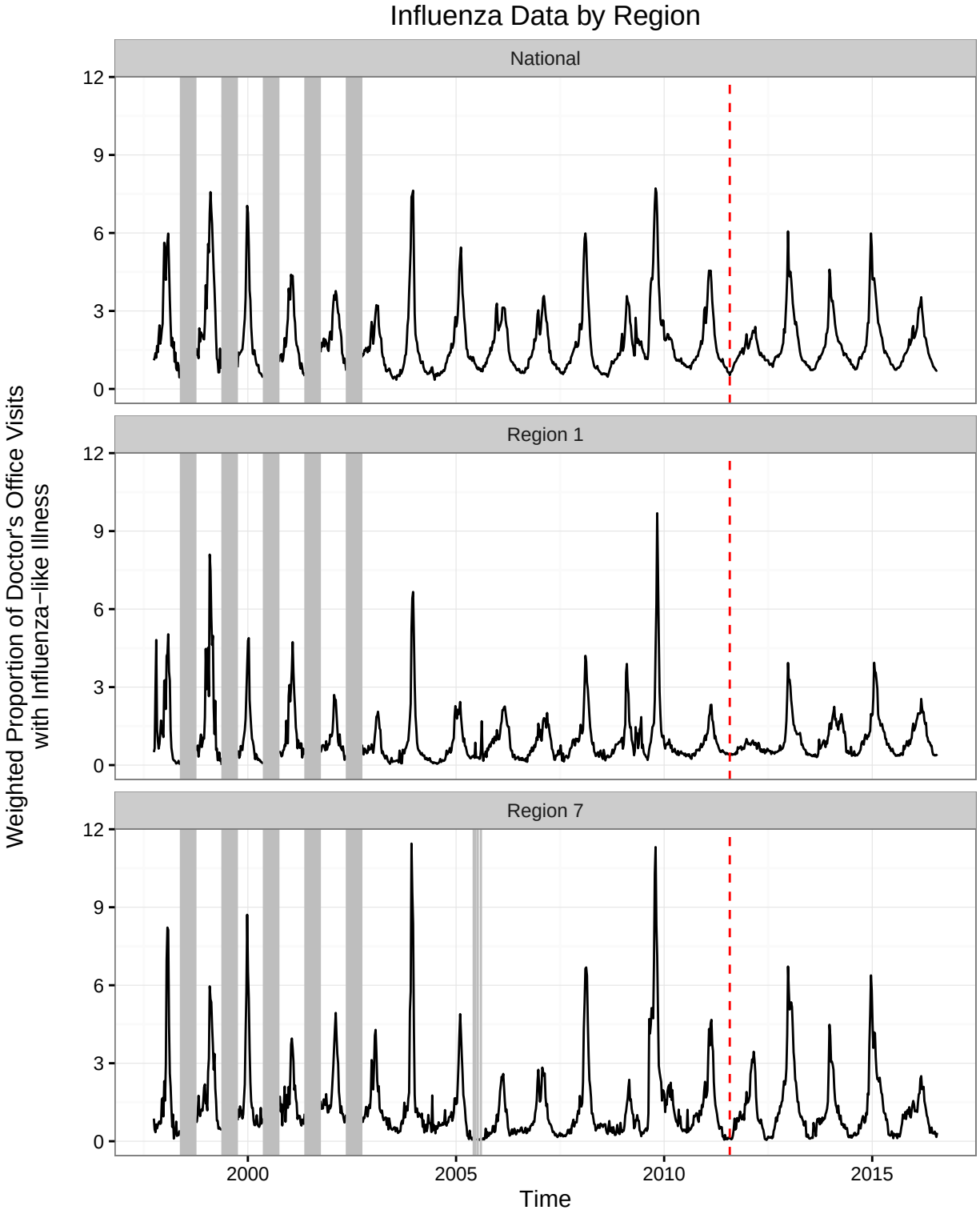

Figure 1: Plot of influenza data. The full data include observations aggregated to the national level and for 10 smaller regions. Here we plot only the data at the national level and in two of the smaller regions; data for the other regions are qualitatively similar. Missing data are indicated with vertical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

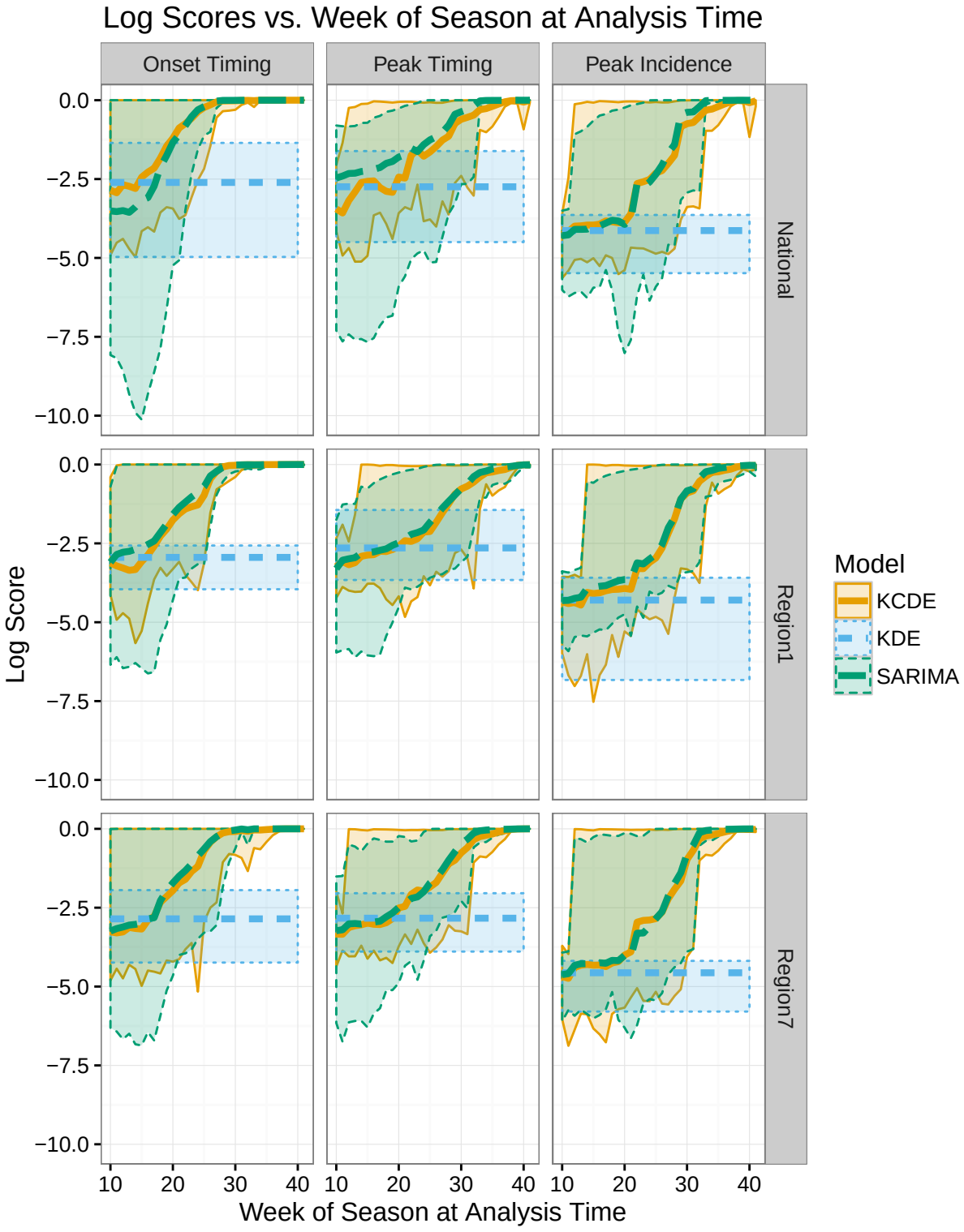

Figure 2: Mean, minimum, and maximum log scores achieved by each component model in each week of the season, summarizing across all seasons in both the train and test phases when all three component models produced predictions.

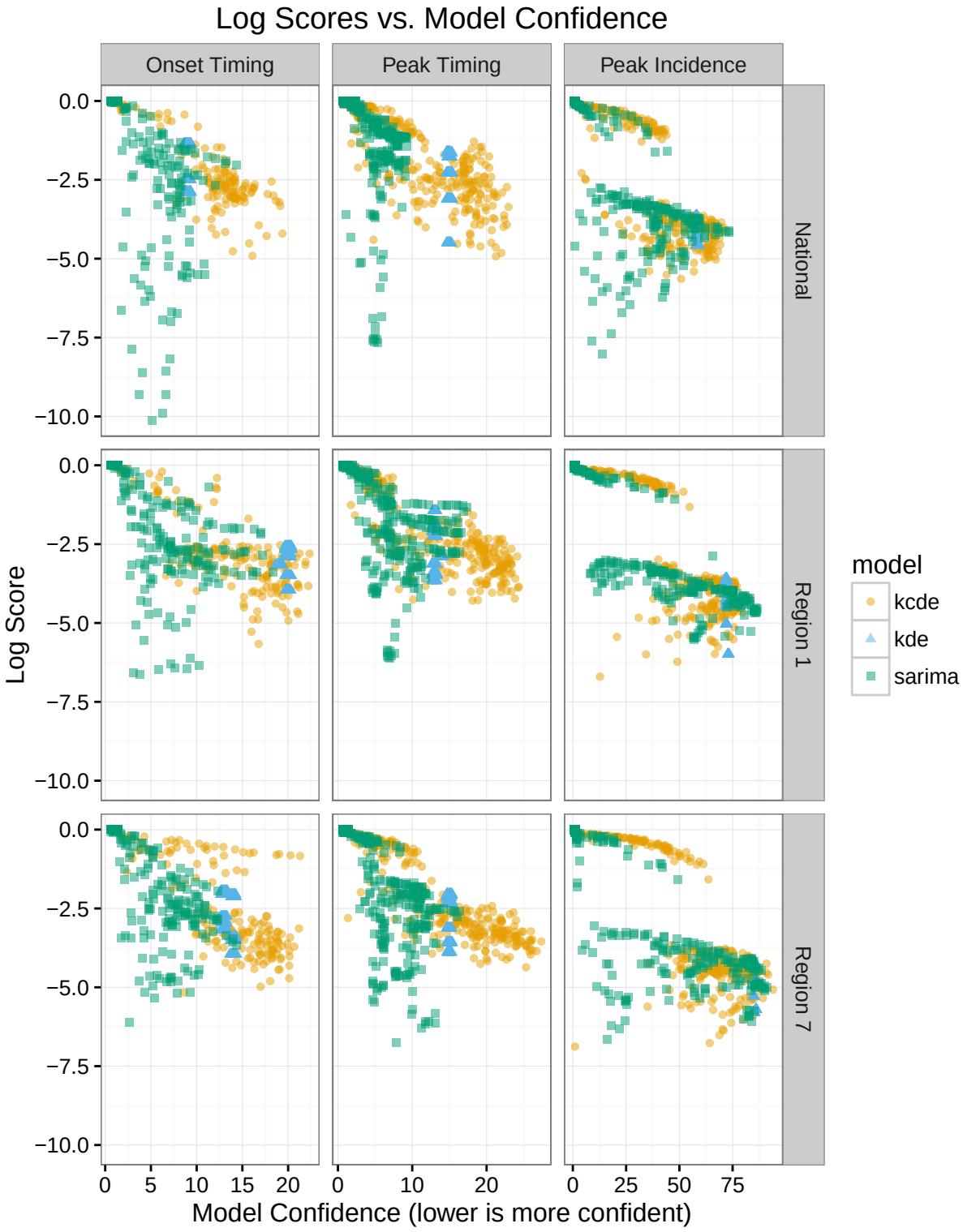

Figure 3: Log scores vs. confidence v1: log scores achieved by each component model vs. model confidence, all seasons in both the train and test phases when all three component models produced predictions.

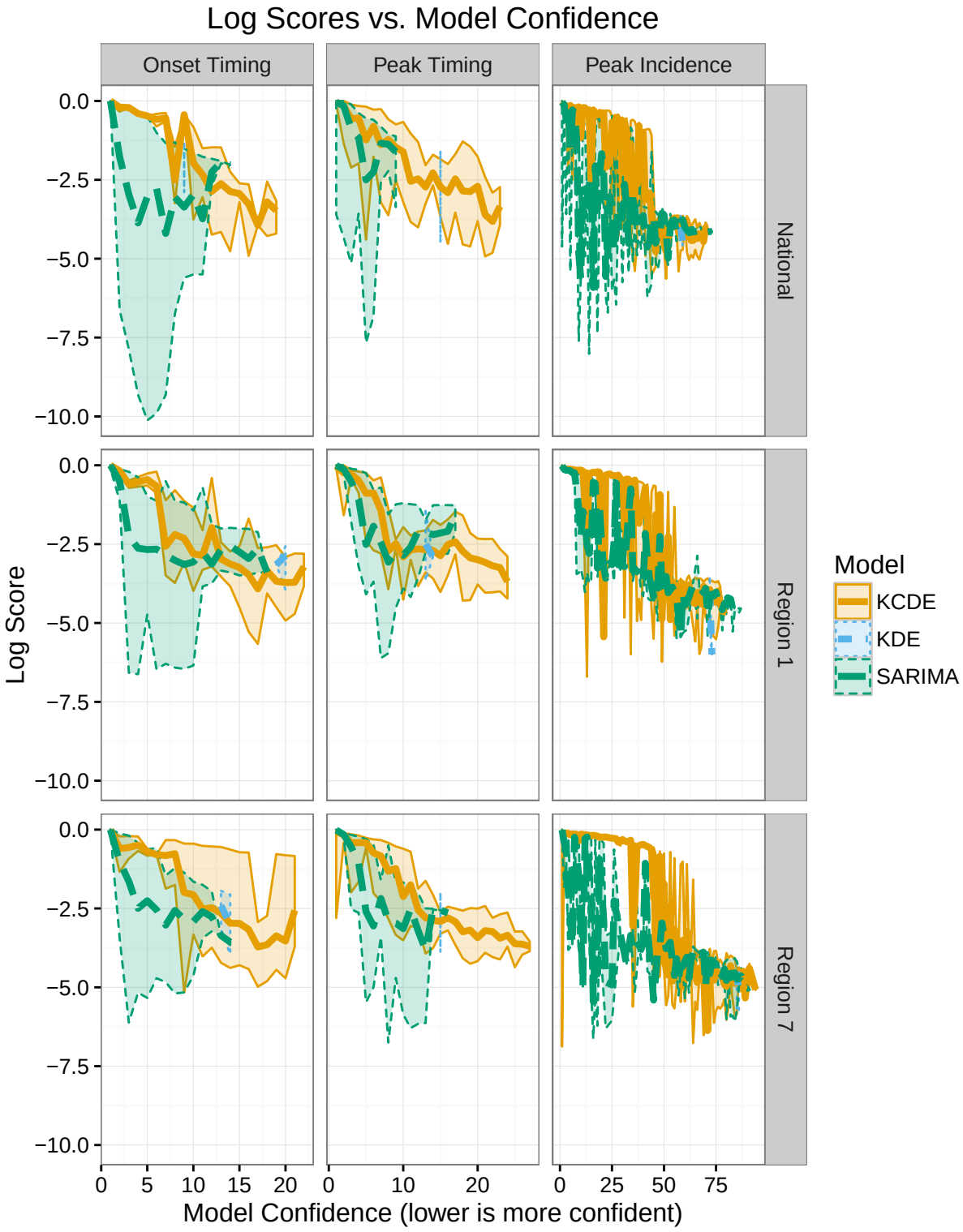

Figure 4: Log scores vs. confidence v2: log scores achieved by each component model vs. model confidence, all seasons in both the train and test phases when all three component models produced predictions.

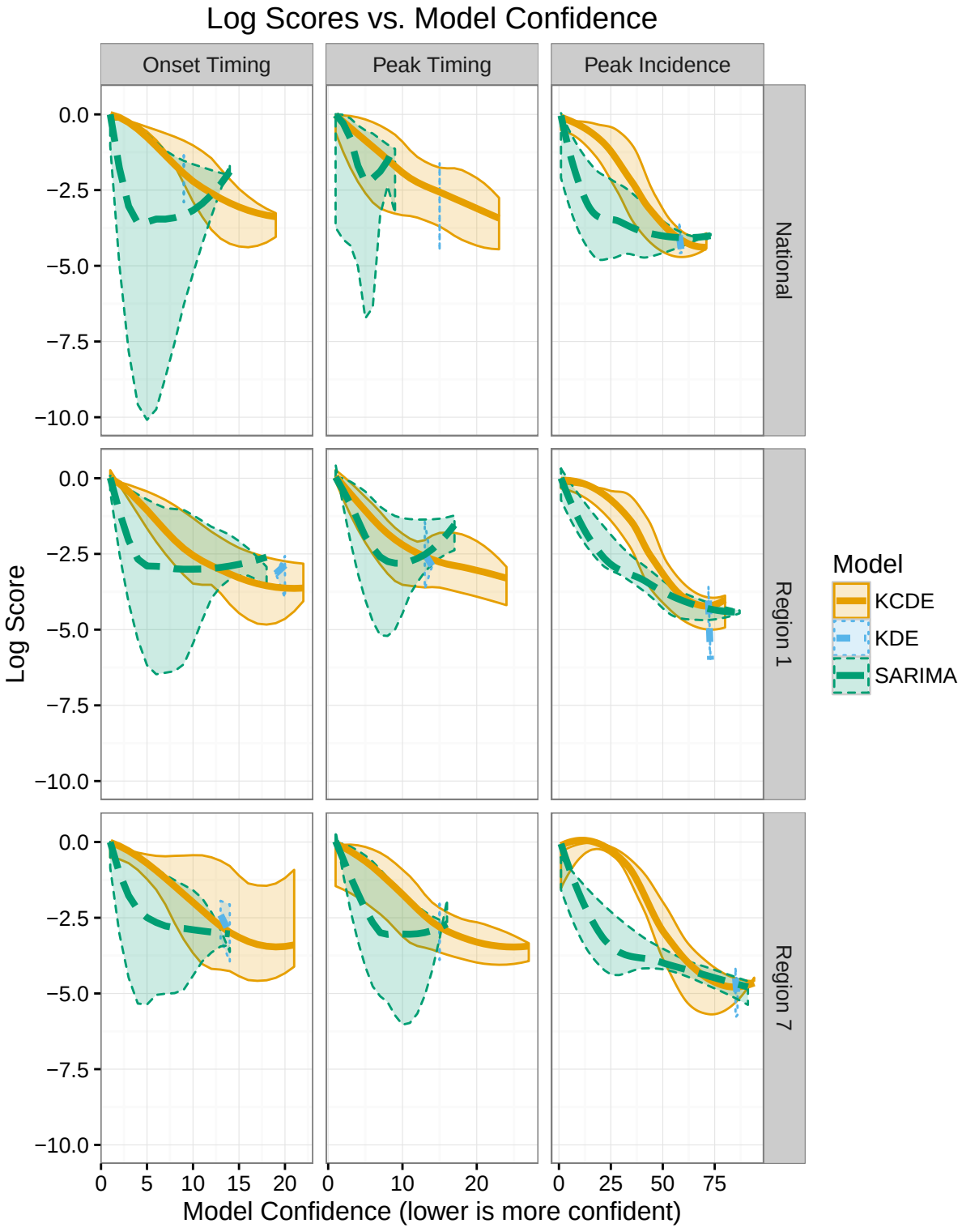

Figure 5: Log scores vs. confidence v3: log scores achieved by each component model vs. model confidence, all seasons in both the train and test phases when all three component models produced predictions.

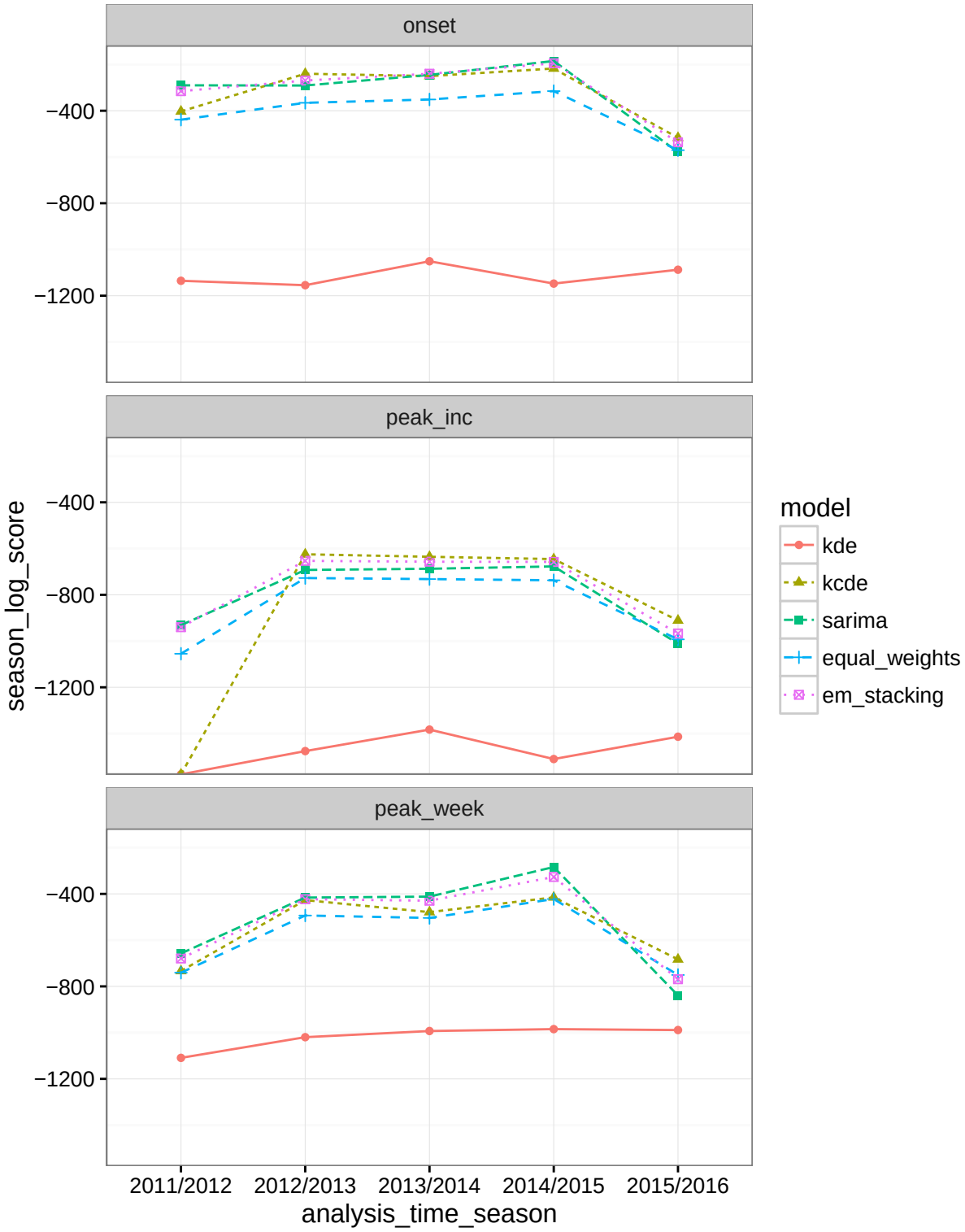

Figure 6: Test phase log scores summary v1: log scores for all predictions in a season summed. Log scores of -Infinity are not represented.

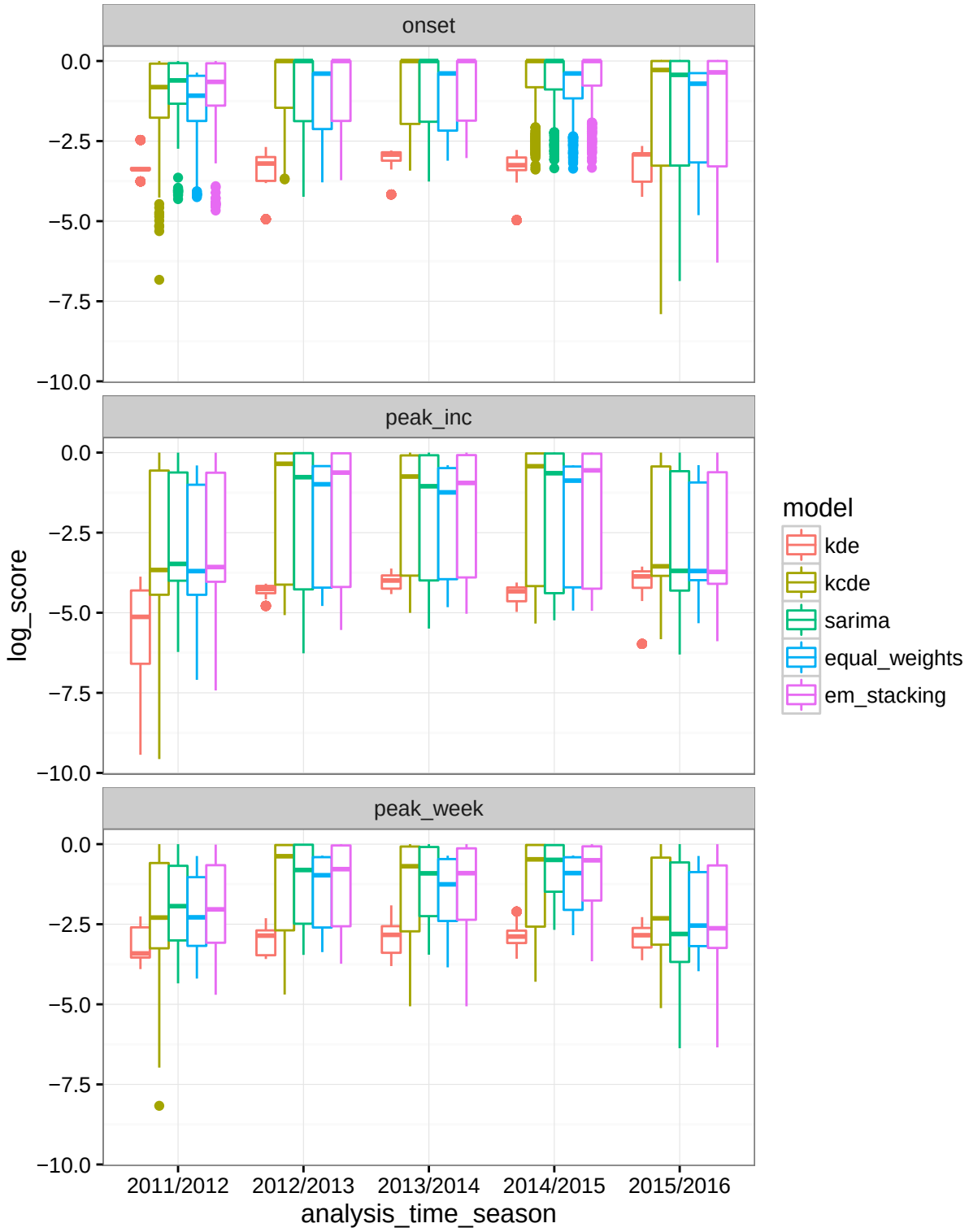

Figure 7: Test phase log scores summary v2: log scores for all predictions in a season summed. Log scores of -Infinity are not represented.

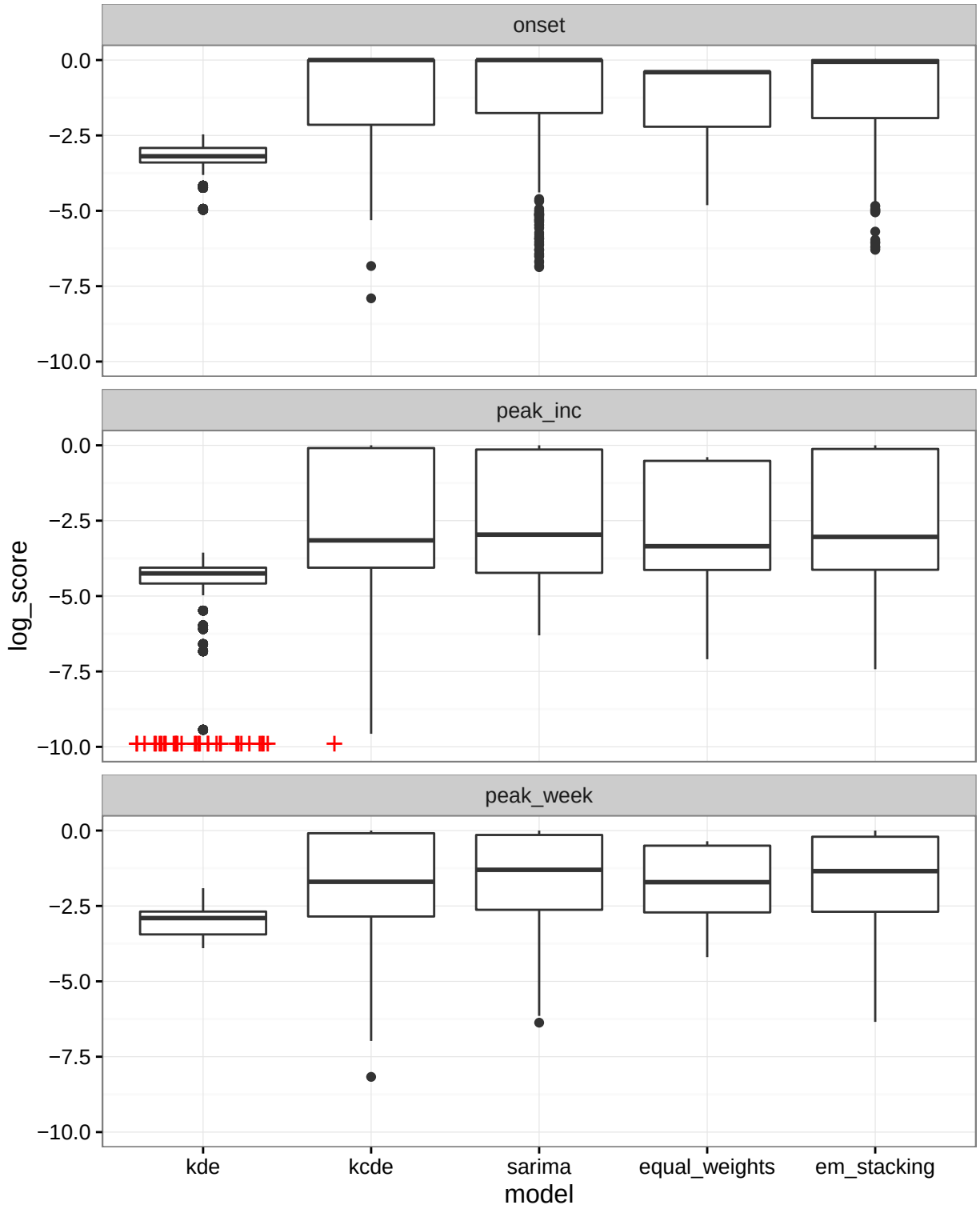

Figure 8: Test phase log scores summary v3. Log scores across all seasons/season weeks represented in box plots. Log scores of -Infinity are represented with a plus sign at -10.

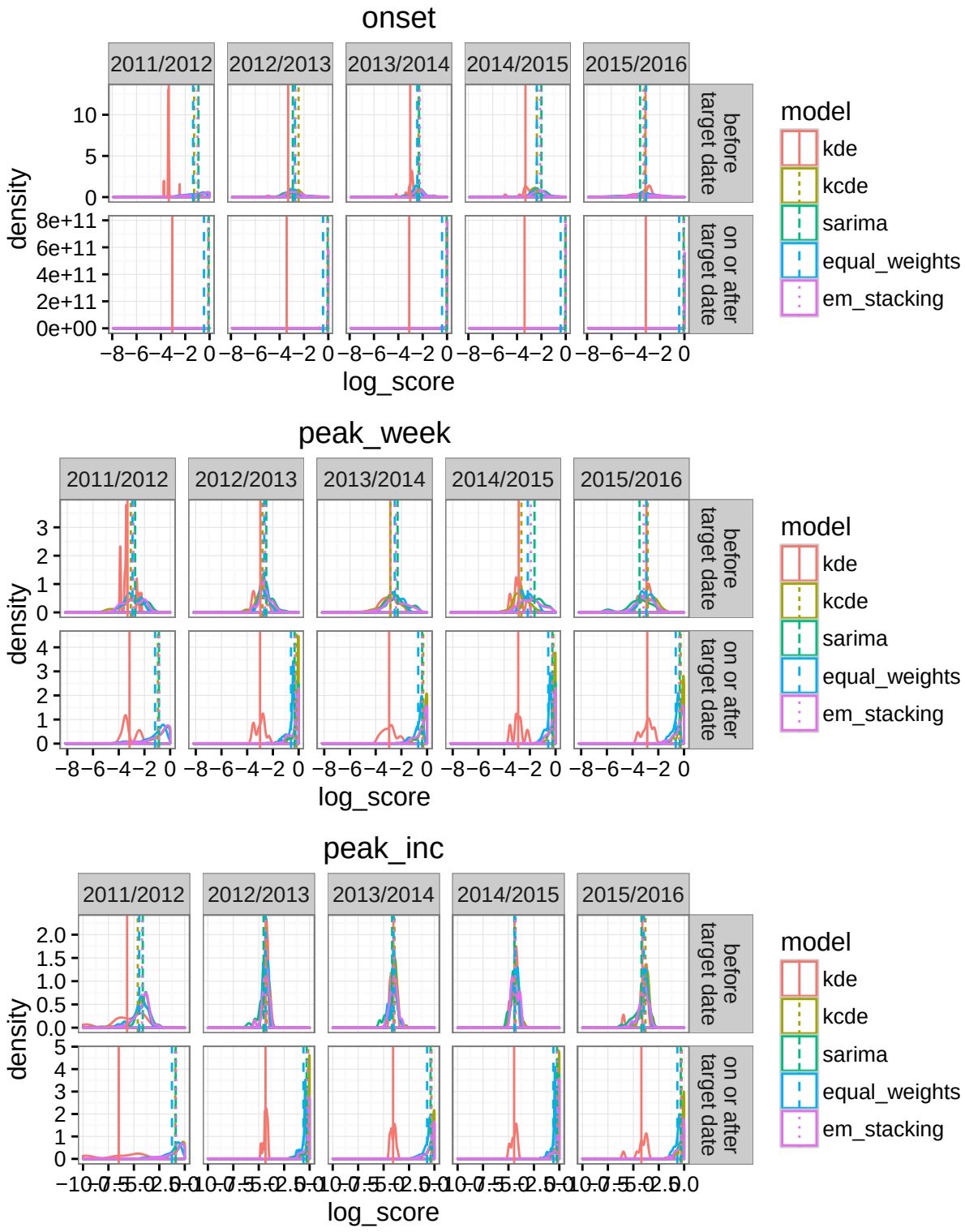

Figure 9: Test phase log scores summary v4. Log scores across all regions and season weeks represented in density plots for each season, split by whether the given season week was before or after the observed onset or peak week. Vertical lines indicate mean log scores, after converting scores of negative infinity to -10.

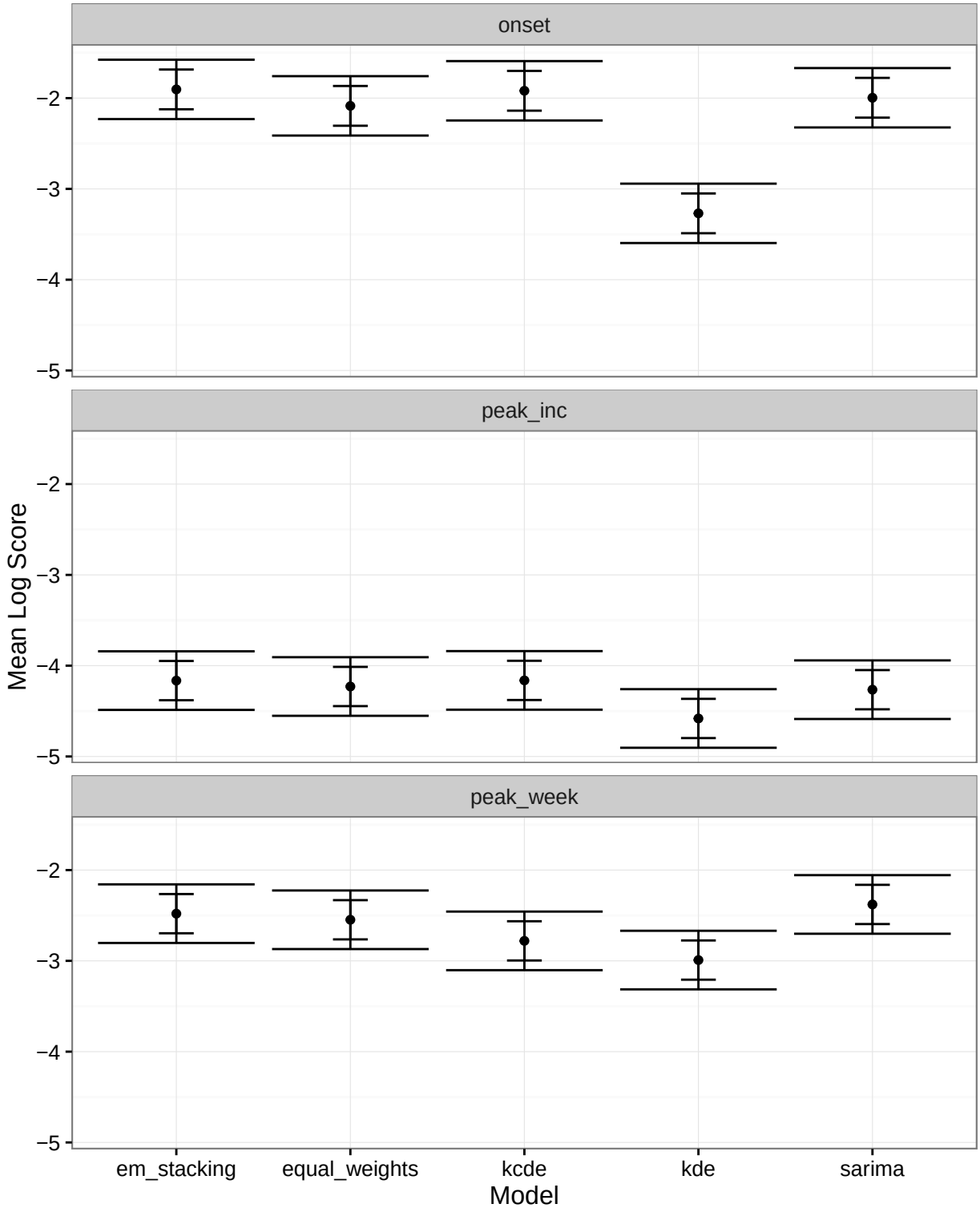

Figure 10: Test phase log scores summary v5. Point estimates and confidence intervals for mean log score for each model in weeks before the target (onset or peak) occurred. Estimates are obtained from a mixed effects model with a separate fixed effect mean for the interaction of model and prediction 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.

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. Overview of Influenza Surveillance in the United States, 2016. 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.

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.