Journal Title
XX(X):1–49

DOI: 10.1177/ToBeAssigned
www.sagepub.com/

Infectious disease prediction with kernel conditional density estimation

Evan L. Ray[1], Krzysztof Sakrejda[1], Stephen A. Lauer[1], Michael Johansen[2] and Nicholas G. Reich[1]

Abstract
Abstract

Keywords
Keywords, Keywords

Introduction

Accurate prediction of infectious disease incidence is important for public health officials planning disease prevention and control measures such as vector control and assignments of medical personnel. Predictive distributions are preferred to point predictions because they communicate uncertainty in the predictions and give decision makers more information in cases where the predictive distribution is skewed or has multiple modes. In this work, we employ a non-parametric approach referred to as kernel conditional density estimation (KCDE) to obtain predictive distributions of disease incidence. When predictions for the trajectory of disease incidence over multiple time horizons are required, our strategy is to use KCDE to obtain marginal predictive distributions for each of those horizons and then combine them using copulas to obtain a joint predictive distribution. In addition to the novel application of these methods to predicting disease incidence, our contributions include a new kernel function that handles both continuous and discrete count data while allowing for a fully parameterized bandwidth matrix and the use of a periodic kernel specification to capture seasonality in disease incidence.

KCDE is a method for estimating the conditional distribution of a random vector $\mathbf{Y}$ given observations of another vector $\mathbf{X}$. In our work, $\mathbf{Y}$ is a measure of disease incidence at some future

[1]Department of Biostatistics and Epidemiology, University of Massachusetts, Amherst
[2]CDC, Puerto Rico

Corresponding author:
Evan Ray, UMass Address Here
Email: elray@umass.edu

Prepared using sagej.cls *[Version: 2015/06/09 v1.01]*

date or dates (the prediction target) and $\mathbf{X}$ is a vector of predictive variables that we condition on in order to make our prediction. For example, $\mathbf{X}$ may include observations of incidence over the most recent few time points, weather covariates, or variables indicating the time of year at which we are making a prediction. KCDE has not previously been applied to obtain predictive distributions in the context of infectious disease, but it has been successfully used for prediction in other settings such as survival time of lung cancer patients[4], female labor force participation[4], bond yields and value at risk in financial markets[3], and wind power[7] among others.

Although KCDE has not previously been applied to predicting infectious disease, closely related methods for obtaining point predictions have been employed for diseases such as measles[12] and influenza[13]. In the infectious disease literature these methods have been referred to as state space reconstruction and the method of analogues, but they amount to an application of nearest neighbors regression methods. The point prediction obtained from nearest neighbors regression is equal to the expected value of the predictive distribution obtained from KCDE if a particular kernel function is used in the formulation of KCDE[6]. However, KCDE offers the advantage of providing a complete predictive distribution rather than only a point prediction. There is also a long history of using other modeling approaches, such as compartmental models, for infectious disease prediction. A full discussion of those methods is beyond the scope of this article; see *** for a recent review.

Double check citation for sugihara – right paper??

Need to find a review of prediction methods for infectious disease.

There is an extensive literature on KCDE, focusing mainly on estimation of continuous conditional densities. Here we offer a brief overview emphasizing the case with mixed continuous and discrete variables; Li and Racine[10] offer a detailed discussion of this case. Throughout this article we use the term density to refer to the Radon-Nikodym derivative of the cumulative distribution function with respect to an appropriately defined measure. In the case of random vectors where some components are continuous random variables and other are discrete, we take this measure to be a product of Lebesgue and counting measures for the corresponding random variables.

Given observations $\{(\mathbf{x}_t, \mathbf{y}_t), t = 1, \ldots, T\}$ the KCDE estimate of the conditional density of $\mathbf{Y}|\mathbf{X}$ is given by

$$\widehat{f}_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \frac{\sum_{t \in \boldsymbol{\tau}} K^{\mathbf{X},\mathbf{Y}} \left\{ (\mathbf{x}', \mathbf{y}')', (\mathbf{x}_t', \mathbf{y}_t')'; \mathbf{H}^{\mathbf{X},\mathbf{Y}} \right\}}{\sum_{t \in \boldsymbol{\tau}} K^{\mathbf{X}}(\mathbf{x}, \mathbf{x}_t; \mathbf{H}^{\mathbf{X}})}. \tag{1}$$

Here, $'$ is the transpose operator and $\boldsymbol{\tau} \subseteq \{1, \ldots, T\}$ indexes the subset of observations used in obtaining the conditional density estimate. In the final density estimate, $\boldsymbol{\tau}$ is typically equal to $\{1, \ldots, T\}$, but proper subsets may be used in the estimation procedures we discuss later. In Equation (1), the numerator is a kernel density estimate of the joint density of $\mathbf{X}$ and $\mathbf{Y}$ and the numerator is a kernel density estimate of the marginal density of $\mathbf{X}$; forming the quotient yields an estimate of the conditional density of $\mathbf{Y}|\mathbf{X}$.

We will work with a slightly restricted specification of Equation (1) in which the kernel function $K^{\mathbf{X},\mathbf{Y}}$ can be written as the product of $K^{\mathbf{X}}$ and a "conditional kernel" $K^{\mathbf{Y}|\mathbf{X}}$:

$$K^{\mathbf{X},\mathbf{Y}} \left\{ (\mathbf{x}', \mathbf{y}')', (\mathbf{x}_t', \mathbf{y}_t')'; \mathbf{H}^{\mathbf{X},\mathbf{Y}} \right\} = K^{\mathbf{X}} \left\{ \mathbf{x}, \mathbf{x}_t; \mathbf{H}^{\mathbf{X}} \right\} K^{\mathbf{Y}|\mathbf{X}} \left\{ \mathbf{y}, \mathbf{y}_t | \mathbf{x}, \mathbf{x}_t; \mathbf{H}^{\mathbf{X},\mathbf{Y}} \right\}. \tag{2}$$

With this restriction, we can rearrange Equation (1) to obtain

$$\widehat{f}_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \sum_{t \in \boldsymbol{\mathcal{T}}} w_t K^{\mathbf{Y}|\mathbf{X}} \left\{ \mathbf{y}, \mathbf{y}_t | \mathbf{x}, \mathbf{x}_t; \mathbf{H}^{\mathbf{X},\mathbf{Y}} \right\}, \text{ where} \tag{3}$$

$$w_t = \frac{K^{\mathbf{X}} \left\{ \mathbf{x}, \mathbf{x}_t; \mathbf{H}^{\mathbf{X}} \right\}}{\sum_{t^* \in \boldsymbol{\mathcal{T}}} K^{\mathbf{X}} \left\{ \mathbf{x}, \mathbf{x}_{t^*}; \mathbf{H}^{\mathbf{X}} \right\}}. \tag{4}$$

We can now interpret $K^{\mathbf{X}}$ as a weighting function determining how much each observation contributes to our final density estimate according to how similar $\mathbf{x}_t$ is to the value $\mathbf{x}$ that we are conditioning on. $K^{\mathbf{Y}|\mathbf{X}}$ is a density function that contributes mass to the final density estimate near the observed value $\mathbf{y}_t$; we require that for any values of $\mathbf{y}_t$, $\mathbf{x}_t$, and $\mathbf{x}$, the integral of $K^{\mathbf{Y}|\mathbf{X}}$ with respect to $\mathbf{y}$ must equal 1. The bandwidth parameters $\mathbf{H}^{\mathbf{X},\mathbf{Y}}$ control the locality and orientation of the weighting function and the contributions to the density estimate from each observation.

In order to complete the formulation of the estimator given in Equation (1), we must specify the kernel functions and describe the procedure that is used to estimate the bandwidth parameters. To our knowledge, all previous authors using kernel methods to estimate either conditional or marginal densities with one or more discrete variables have employed a kernel function that is a product of univariate kernel functions [1,?,8,11,]. For the continuous variables, there are several options for the functional form; one common choice is the Gaussian kernel. Several alternatives have also been proposed for use with discrete variables. For ordered discrete variables, symmetric univariate kernel functions that decrease with the distance between x and x_t are often used; several specific functional forms that achieve this have been used in the literature [1;8;9;11?] . For count data, alternatives that have been proposed include.

Using a product kernel simplifies the mathemetical formulation of the kernel function when both continuous and discrete variables are present, but has the effect of forcing the kernel function to be orientied in line with the coordinate axes. In settings with only continuous variables, it is common to use a multivariate kernel function with a bandwidth parameterization that allows for orientations in directions other than along the coordinate axes. Asymptotic analysis and experience with applications have shown that this can result in improved density estimates in many cases (cite ***). One possibility for introducing orientation to the kernel function is to use a fully parameterized bandwidth matrix, allowing the kernel to be oriented in any direction. Another common alternative is to fix the orientation to be in the directions of the eigenvectors of the sample covariance matrix.

Estimation. Two main strategies: cross validation and rule-based. Targets for optimization in cross-validation. For estimating joint densities without conditioning, [5] have shown that with dependent data, but small gains in the mean integrated square error of the density estimate relative to the true conditional density can be achieved by leaving out observations adjacent to the time point whose density is being estimated.

A limitation of kernel-based density estimation methods is that they may not scale well with the dimension of the vector whose distribution is being estimated. This is particularly relevant in our application, where it is desired to obtain joint predictive distributions for disease incidence over the course of many weeks. Copulas present one strategy for estimating the joint distribution

of moderate to high dimensional random vectors, and work by specifying a relatively simple parametric model for the dependence relations among those variables. Specifically, we model the joint distribution of $Y_1, \ldots, Y_D$ by $F_{Y_1,\ldots,Y_D}(y_1, \ldots, y_D) = c(F_{Y_1}(y_1), \ldots, F_{Y_D}(y_D); \boldsymbol{\theta}_c)$. Here $c : [0,1]^D \to [0,1]$ is the copula function depending on parameters $\boldsymbol{\theta}_c$ and mapping the vector of marginal c.d.f. values to the joint c.d.f. value.

The remainder of this article is organized as follows. We: - describe how kernel density estimation with a non-diagonal bandwidth can be achieved using a partially discretized multivariate normal distribution for the kernel functions.

- simulation study comparing product and non-product formulations for marginal and conditional density estimation

- applications

Method Description

Suppose we observe $\mathbf{z}_t = \{z_{t,1}, \ldots, z_{t,D}\} \in \mathbb{R}^D$ at each point in time $t = 1, \ldots, T$. Our goal is to obtain a predictive distribution for one of the observed variables, with index $d_{pred} \in \{1, \ldots, D\}$, over a range of prediction horizons contained in the set $\mathcal{P}$. For example, if we have weekly data and we are interested in obtaining predictions for a range between 4 and 6 weeks after the most recent observation then $\mathcal{P} = \{4, 5, 6\}$. Let P be the largest element of the set $\mathcal{P}$ of prediction horizons.

For each time $t \in \boldsymbol{\tau}$, we form the vectors $\mathbf{y}_t$ and $\mathbf{x}_t$ representing the prediction target and predictive variables respectively.

In order to perform prediction, we will use lagged observations. Let $\mathbf{l}^{max} = (l_1^{max}, \ldots, l_D^{max})$ specify the maximum number of lags for each observed variable that may be used for prediction, and let $L = \max dl_d^{max}$ be the overall largest lag that may be used across all variables. In the estimation procedure we describe in Section , we will select a subset of these lags to actually use in the predictions. We capture which lags are actually used in the vector

$$\mathbf{u} = (u_{1,0}, \ldots, u_{1,l_1^{max}}, \ldots, u_{D,0}, \ldots, u_{D,l_D^{max}}), \text{ where}$$
$$u_{d,l} = \begin{cases} 0 \text{ if lag } l \text{ of variable } d \text{ is not used in forming predictions} \\ 1 \text{ if lag } l \text{ of variable } d \text{ is used in forming predictions.} \end{cases}$$

By analogy with the standard notation in autoregressive models, we define

$$\mathbf{y}_t = (z_{t,d_{pred}}, \ldots, B^{(P-1)} z_{t,d_{pred}}) \text{ and}$$
$$\mathbf{x}_t = (B^{(P)} z_{t,1}, \ldots, B^{(P+l_1^{max}-1)} z_{t,1}, \ldots, B^{(P)} z_{t,D}, \ldots, B^{(P+l_D^{max}-1)} z_{t,D})$$

Here, $B^{(a)}$ is the backshift operator defined by $B^{(a)} z_{t,d} = z_{t-a,d}$. Note that the lengths of $\mathbf{y}_t$ and $\mathbf{x}_t$, as well as exactly which lags are used to form them, depend on $\mathcal{P}$ and $\mathbf{l}^{max}$; we suppress this dependence in the notation for the sake of clarity. The vector $\mathbf{y}_t$ represents the prediction target when our most recent observation was made at time $t - P$: the vector of observed values at each prediction horizon $p \in \mathcal{P}$. The variable $\mathbf{x}_t$ represents the vector of all lagged covariates that are available for use in performing prediction.

To make the notation concrete, suppose that $\mathbf{z}_t$ contains the observed case count for week t in San Juan, the observed case count for week t in Iquitos, and the date on Monday of week t, and our goal is to predict the weekly case count in San Juan. Then $D = 3$ and $d_{pred} = 1$. If we want to predict the weekly case counts for the two weeks after the most recently observation, then $p = 2$. If we specify that the model may include the two most recent observations for the case counts in San Juan and Iquitos, but only the time index at the most recent observation then $\mathbf{l}^{max} = (1, 1, 0)$. If our current model uses only the most recently observed case counts for San Juan and Iquitos then $\mathbf{u} = (1, 0, 1, 0, 0)$, where the 1's are in the positions of the $\mathbf{u}$ vector representing lag 0 of the counts for San Juan and lag 0 of the counts for Iquitos. The variable $y_t^{(P)}$ is a vector containing the observed case counts for San Juan in weeks $t + 1$ and $t + 2$; $\mathbf{x}_t^{(\mathbf{l}^{max})}$ contains the observed case counts for San Juan and Iquitos in weeks t and $t - 1$ as well as the time index variable in week t.

In order to perform prediction, we regard $\{(\mathbf{y}_t, \mathbf{x}_t), t = 1 + P + L, \ldots, T\}$ as a sample from the joint distribution of $(\mathbf{Y}, \mathbf{X})$. We wish to estimate the conditional distribution of $\mathbf{Y}|\mathbf{X}$. In order to do this, we employ kernel density estimation. Let $K^{\mathbf{Y}}(\mathbf{y}, \mathbf{y}^*, H^{\mathbf{Y}})$ and $K^{\mathbf{X}}(\mathbf{x}, \mathbf{x}^*, H^{\mathbf{X}})$ be kernel functions centered at $\mathbf{y}^*$ and $\mathbf{x}^*$ respectively and with bandwidth matrices $H^{\mathbf{Y}}$ and $H^{\mathbf{X}}$. We estimate the conditional distribution of $\mathbf{Y}|\mathbf{X}$ as follows:

$$\widehat{f}_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{X} = \mathbf{x}) = \frac{\widehat{f}_{\mathbf{Y},\mathbf{X}}(\mathbf{y}, \mathbf{x})}{\widehat{f}_{\mathbf{X}}(\mathbf{x})} \tag{5}$$

$$= \frac{\sum_{t \in \tau} K^{\mathbf{Y},\mathbf{X}}\{(\mathbf{y}, \mathbf{x}), (\mathbf{y}_t, \mathbf{x}_t), H^{\mathbf{Y},\mathbf{X}}\}}{\sum_{t \in \tau} K^{\mathbf{X}}(\mathbf{x}, \mathbf{x}_t, H^{\mathbf{X}})} \tag{6}$$

$$= \frac{\sum_{t \in \tau} K^{\mathbf{Y}|\mathbf{X}}(\mathbf{y}, \mathbf{y}_t|\mathbf{x}, \mathbf{x}_t, H^{\mathbf{Y},\mathbf{X}}) K^{\mathbf{X}}(\mathbf{x}, \mathbf{x}_t, H^{\mathbf{X}})}{\sum_{t \in \tau} K^{\mathbf{X}}(\mathbf{x}, \mathbf{x}_t, H^{\mathbf{X}})} \tag{7}$$

$$= \sum_{t \in \tau} w_t K^{\mathbf{Y}|\mathbf{X}}(\mathbf{y}, \mathbf{y}_t|\mathbf{x}, \mathbf{x}_t, H^{\mathbf{Y},\mathbf{X}}), \text{ where} \tag{8}$$

$$w_t = \frac{K^{\mathbf{X}}(\mathbf{x}, \mathbf{x}_t, H^{\mathbf{X}})}{\sum_{t^* \in \tau} K^{\mathbf{X}}(\mathbf{x}, \mathbf{x}_{t^*}, H^{\mathbf{X}})} \tag{9}$$

In Equation (5), we are making use of the fact that the conditional density for $\mathbf{Y}|\mathbf{X}$ can be written as the quotient of the joint density for $(\mathbf{Y}, \mathbf{X})$ and the marginal density for $\mathbf{X}$. In Equation (7), we obtain separate kernel density estimates for the joint and marginal densities in this quotient. In Equation (8), we rewrite this quotient by passing the denominator of Equation (7) into the summation in the numerator. We can interpret the result as a weighted kernel density estimate, where each observation $t \in \tau$ contributes a different amount to the final conditional density estimate. The amount of the contribution from observation t is given by the weight w_t, which effectively measures how similar $\mathbf{x}_t$ is to the point $\mathbf{x}$ at which we are estimating the conditional density. If $\mathbf{x}_t^{(\mathbf{l}^{max})}$ is similar to $\mathbf{x}_{t^*}^{(\mathbf{l}^{max})}$, a large weight is assigned to t; if $\mathbf{x}_t^{(\mathbf{l}^{max})}$ is different from $\mathbf{x}_{t^*}^{(\mathbf{l}^{max})}$, a small weight is assigned to t.

In kernel density estimation, it is generally required that the kernel functions integrate to 1 in order to obtain valid density estimates. However, after conditioning on $\mathbf{X}$, it is no longer

necessary that $K^{\mathbf{X}}(\mathbf{x}, \mathbf{x}_t, H^{\mathbf{X}})$ integrate to 1. In fact, as can be seen from Equation (9), any multiplicative constants of proportionality will cancel out when we form the observation weights. We can therefore regard $K^{\mathbf{X}}(\mathbf{x}, \mathbf{x}_t, H^{\mathbf{X}})$ as a more general weighting function that measures the similarity between $\mathbf{x}$ and $\mathbf{x}_t$. As we will see, eliminating the constraint that $K^{\mathbf{X}}$ integrates to 1 is a useful expansion the space of functions that can be used in calculating the observation weights. However, we still require that $K^{\mathbf{Y}}$ integrates to 1.

In Equations (5) through (9), τ is an index set of time points used in obtaining the density estimate. In most settings, we can take $\tau = \{1 + P + L, \ldots, T\}$. These are the time points for which we can form the lagged observation vector $\mathbf{x}_t$ and the prediction target vector $\mathbf{y}_t$. However, we will place additional restrictions on the time points included in τ in the cross-validation procedure discussed in Section .

If we wish to obtain point predictions, we can use a summary of the predictive density. For example, if we take the expected value, we obtain kernel regression:

$$(\widehat{\mathbf{Y}}|\mathbf{X} = \mathbf{x}) = \mathbb{E}_{\widehat{f}_{\mathbf{Y}|\mathbf{X}}}\{\mathbf{Y}|\mathbf{X} = \mathbf{x}\} \tag{10}$$

$$= \int \sum_{t \in \tau} w_t K^{\mathbf{Y}}(\mathbf{y}, \mathbf{y}_t, H^{\mathbf{Y}}) \mathbf{y} \, d\mathbf{y} \tag{11}$$

$$= \sum_{t \in \tau} w_t \mathbf{y}_t \tag{12}$$

The equality in Equation (12) holds if the kernel function $K^{\mathbf{Y}}(\mathbf{y}, \mathbf{y}_t, H^{\mathbf{Y}})$ is symmetric about $\mathbf{y}_t$, or more generally if it is the pdf of a random variable with expected value $\mathbf{y}_t$.

Another alternative that we pursue is the use of smoothed observations in forming the lagged observation vectors. We use smoothed case counts on a log scale for the weighting kernels, and the unsmoothed case counts on the original scale for the prediction kernels.

Parameter Estimation

We use cross-validation to select the variables that are used in the model and estimate the corresponding bandwidth parameters by (approximately) minimizing a cross-validation measure of the quality of the predictions obtained from the model. Formally,

$$(\widehat{\mathbf{u}}, \widehat{H}^{\mathbf{X}}, \widehat{H}^{\mathbf{Y}}) \approx \underset{(\mathbf{u}, H^{\mathbf{X}}, H^{\mathbf{Y}})}{\operatorname{argmin}} \sum_{t^*=1+P+L}^{T} Q[\mathbf{y}_{t^*}, \widehat{f}(\mathbf{y}|\mathbf{X} = \mathbf{x}_{t^*}; \mathbf{u}, H^{\mathbf{X}}, H^{\mathbf{Y}}, \{(\mathbf{y}_t, \mathbf{x}_t) : t \in \tau_{t^*}\})] \tag{13}$$

Here, Q is a loss function that measures the quality of the estimated density $\widehat{f}$ given an observation $\mathbf{y}_{t^*}$. We have made the dependence of this estimated density on the the parameters $\mathbf{u}$, $H^{\mathbf{x}}$, and $H^{\mathbf{Y}}$, as well as on the data $\{(\mathbf{y}_t, \mathbf{x}_t) : t \in \tau_{t^*}\}$, explicit in the notation. In order to reduce the potential for our parameter estimates to be affected by local correlation in the time series, we eliminate all time points that fall within one year of t^* from the index set τ_{t^*} used to form the conditional density estimate $\widehat{f}(\mathbf{y}|\mathbf{X} = \mathbf{x}_{t^*}; \mathbf{u}, H^{\mathbf{X}}, H^{\mathbf{Y}}, \{(\mathbf{y}_t, \mathbf{x}_t) : t \in \tau_{t^*}\})$.

Talk about proper scoring rules and our particular choice of Q.

We use a forward/backward stagewise procedure to obtain the set of combinations of variables and lags that are included in the final model (represented by $\mathbf{u}$). For each candidate model, we use the limited memory box constrained optimization procedure of[?] to estimate the bandwidth parameters. The approximation in Equation (13) is due to the fact that this optimization procedure may not find a global minimum.

Simulation Studies

In this Section, we conduct two sets of simulation studies designed to answer two separate questions:

1. How much does using a kernel function with a non-diagonal bandwidth matrix contribute to the quality of conditional density estimates relative to density estimates obtained through KCDE using diagonal bandwidth matrices?
2. How does our method perform in the context of seasonal time series data? Specifically, how does the method perform relative to common alternatives, and how much do each of our three contributions (non-diagonal bandwidth matrices for discrete data, using a periodic function of time as predictive variable, and use of low band-pass filtered observatiosn as predictive variables) contribute to predictive performance?

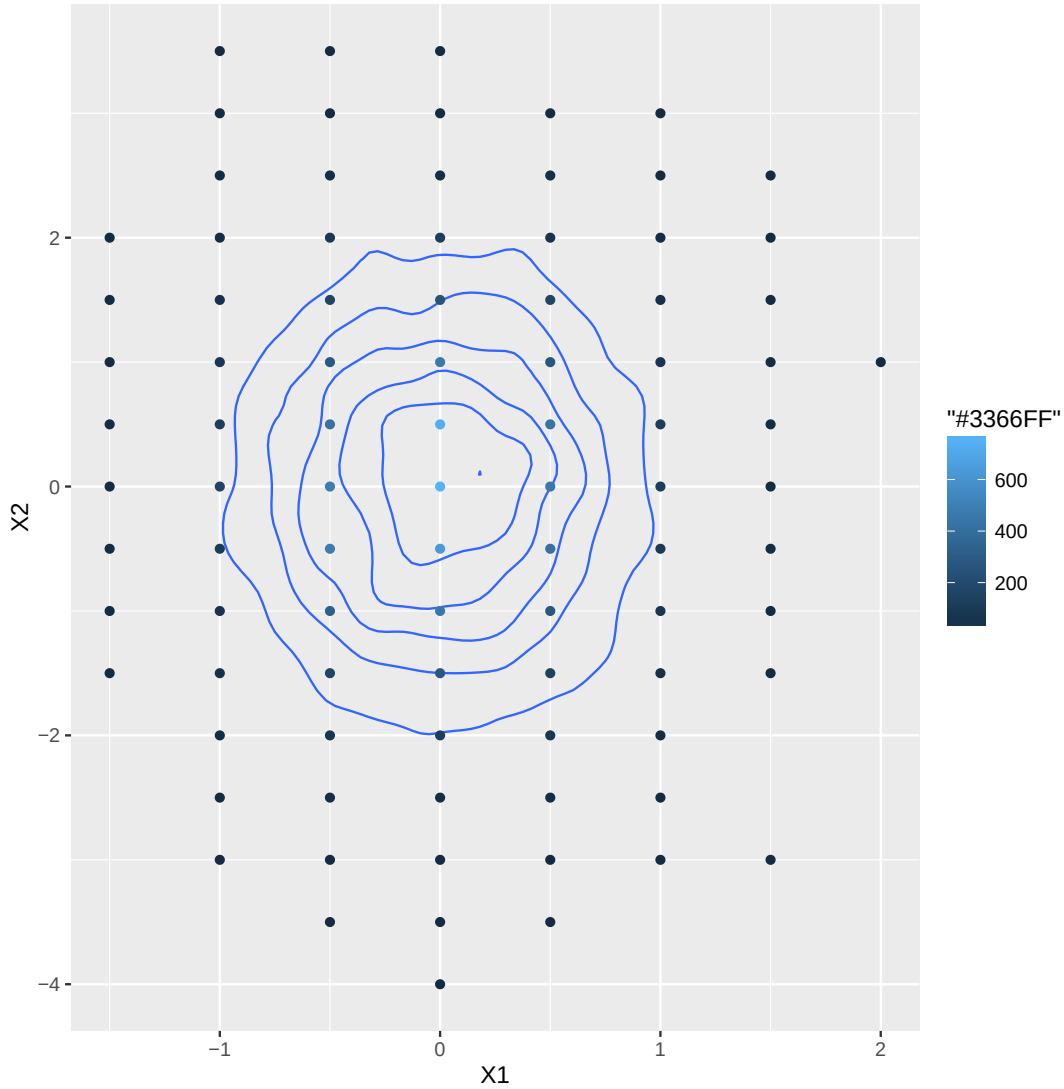
Comparison of KCDE approaches

Our first set of simulation studies is based closely on those conducted in[2]; their examples demonstrate the utility of using a fully parameterized bandwidth matrix in kernel density estimation of continuous distributions. We modify their simulation study to examine the benefits of fully parameterized bandwidth matrices in the context of conditional density estimation with discrete variables.

We simulate observations from each of seven distributions. The first five of these are plotted in Figure ***.

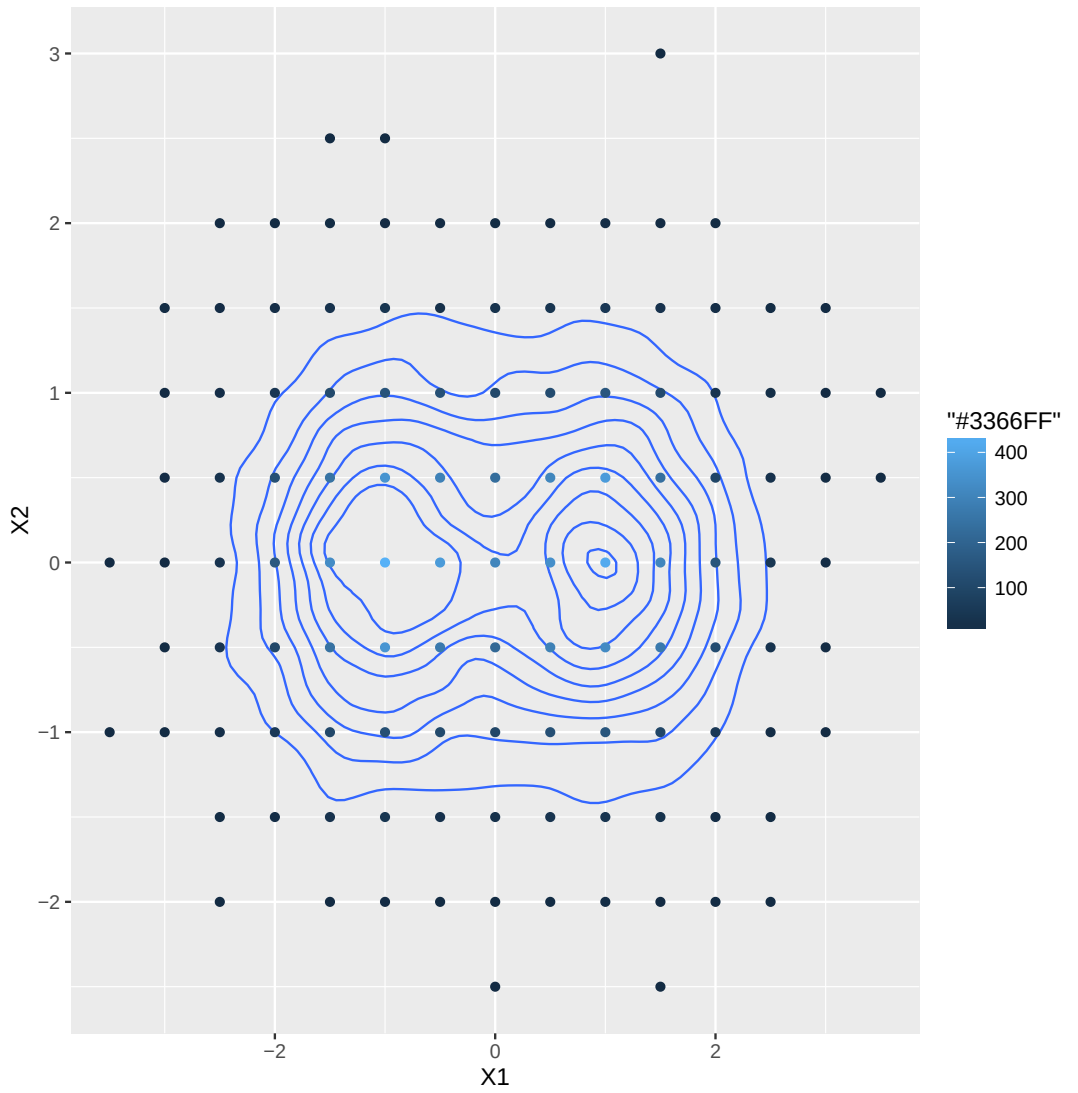
```
library(ggplot2)
library(grid)
library(plyr)
library(dplyr)
library(tidyr)
library(pdtmvn)
library(kcde)
source("/media/evan/data/Reich/infectious-disease-prediction-with-kcde/inst/code/sim-densi

## Density family bivariate-A
n_sim <- 10000
discrete_sample <- sim_from_pdtmvn_mixt(n = n_sim, sim_family = "bivariate-A-discretized")
    as.data.frame()
continuous_sample <- sim_from_pdtmvn_mixt(n = n_sim, sim_family = "bivariate-A") %>%
```

```
    as.data.frame()
discrete_sample_counts <- discrete_sample %>%
    count(X1, X2)

pa <- ggplot() +
    geom_density_2d(aes(x = X1, y = X2), data = continuous_sample) +
    geom_point(aes(x = X1, y = X2, colour = n), data = discrete_sample_counts)
pa
```

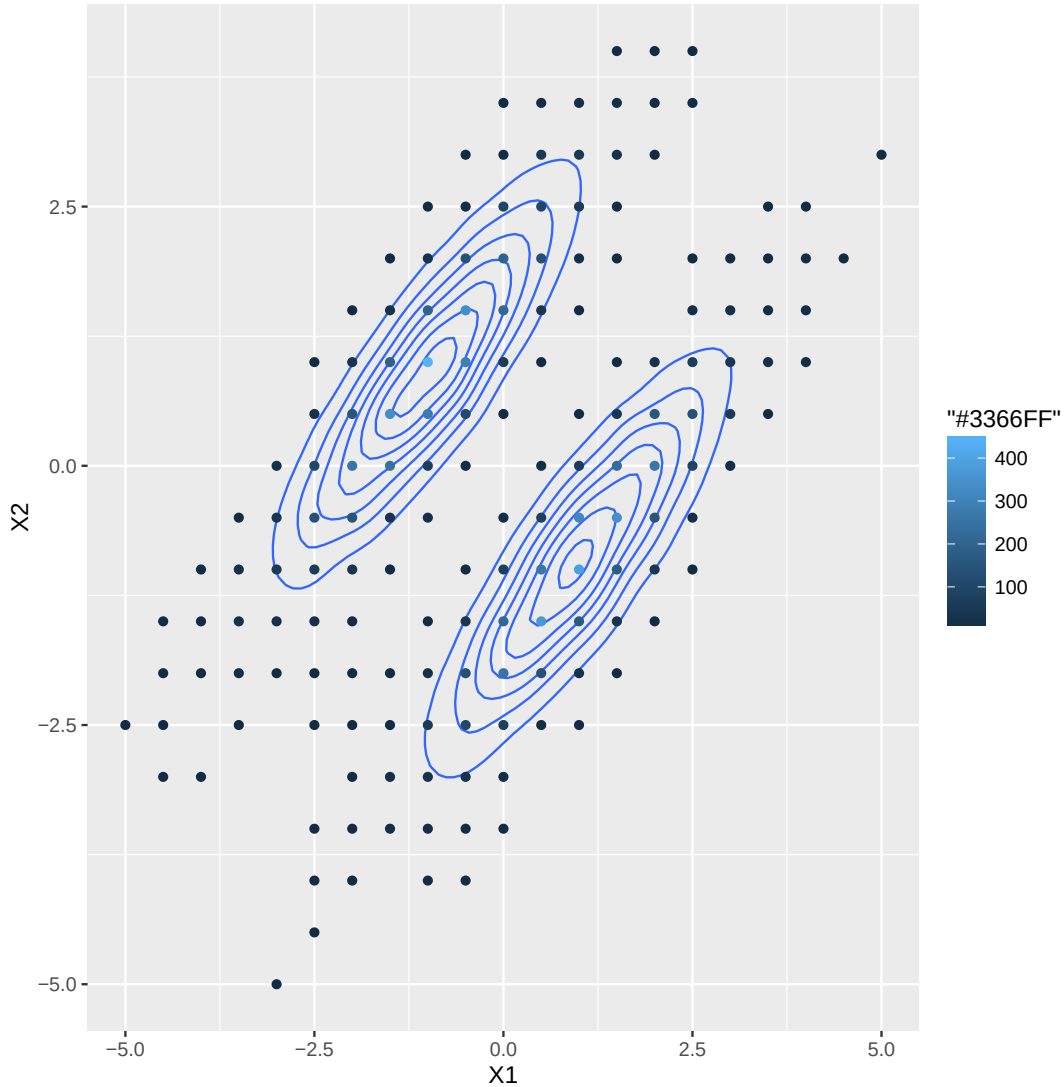



```
## Density family bivariate-B
n_sim <- 10000
discrete_sample <- sim_from_pdtmvn_mixt(n = n_sim, sim_family = "bivariate-B-discretized")
    as.data.frame()
continuous_sample <- sim_from_pdtmvn_mixt(n = n_sim, sim_family = "bivariate-B") %>%
    as.data.frame()
discrete_sample_counts <- discrete_sample %>%
```

```
    count(X1, X2)

pb <- ggplot() +
    geom_density_2d(aes(x = X1, y = X2), data = continuous_sample) +
    geom_point(aes(x = X1, y = X2, colour = n), data = discrete_sample_counts)
pb
```

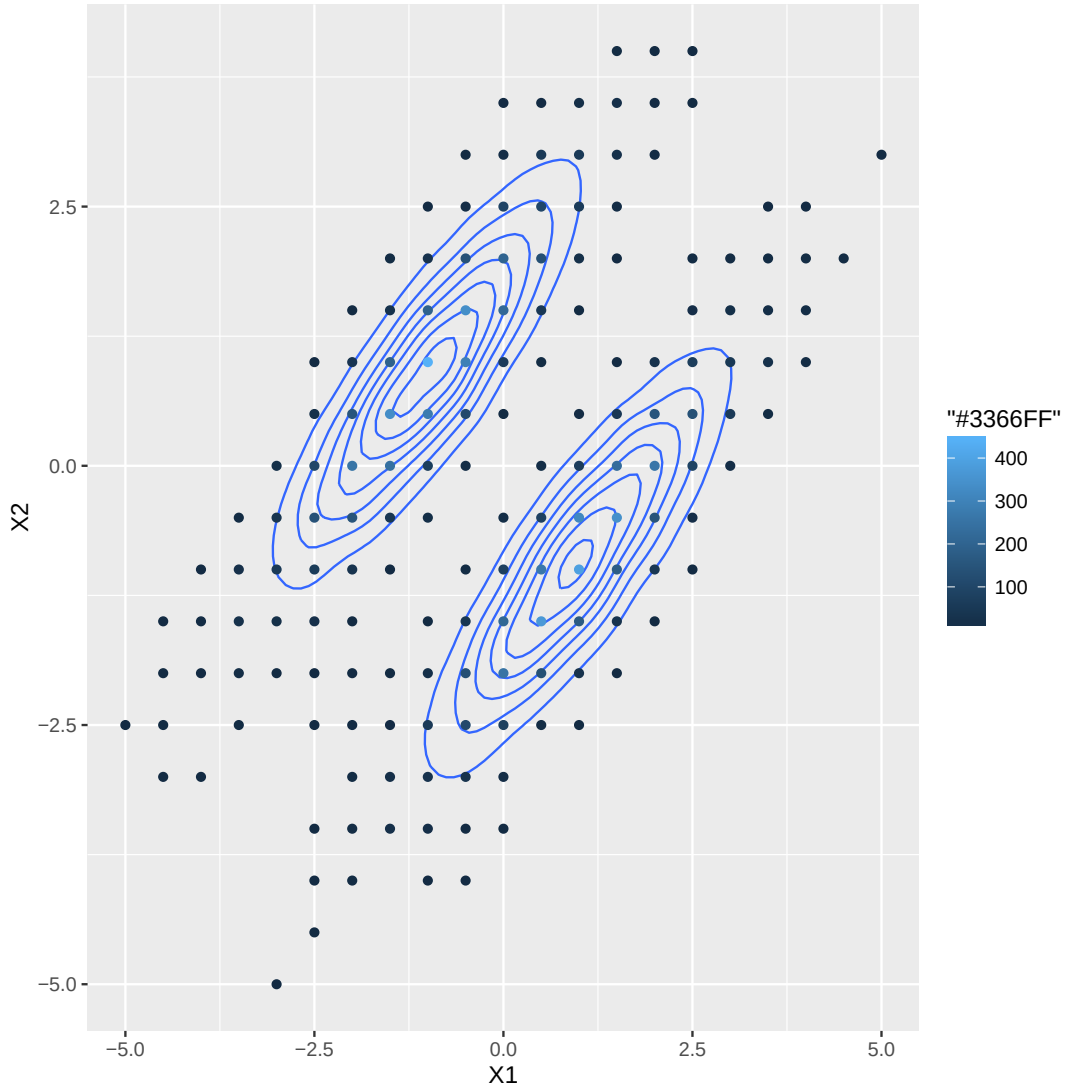


```r
## Density family bivariate-C
n_sim <- 10000
discrete_sample <- sim_from_pdtmvn_mixt(n = n_sim, sim_family = "bivariate-C-discretized") %>%
    as.data.frame()
continuous_sample <- sim_from_pdtmvn_mixt(n = n_sim, sim_family = "bivariate-C") %>%
    as.data.frame()
discrete_sample_counts <- discrete_sample %>%
    count(X1, X2)

pc <- ggplot() +
    geom_density_2d(aes(x = X1, y = X2), data = continuous_sample) +
    geom_point(aes(x = X1, y = X2, colour = n), data = discrete_sample_counts)
pc
```



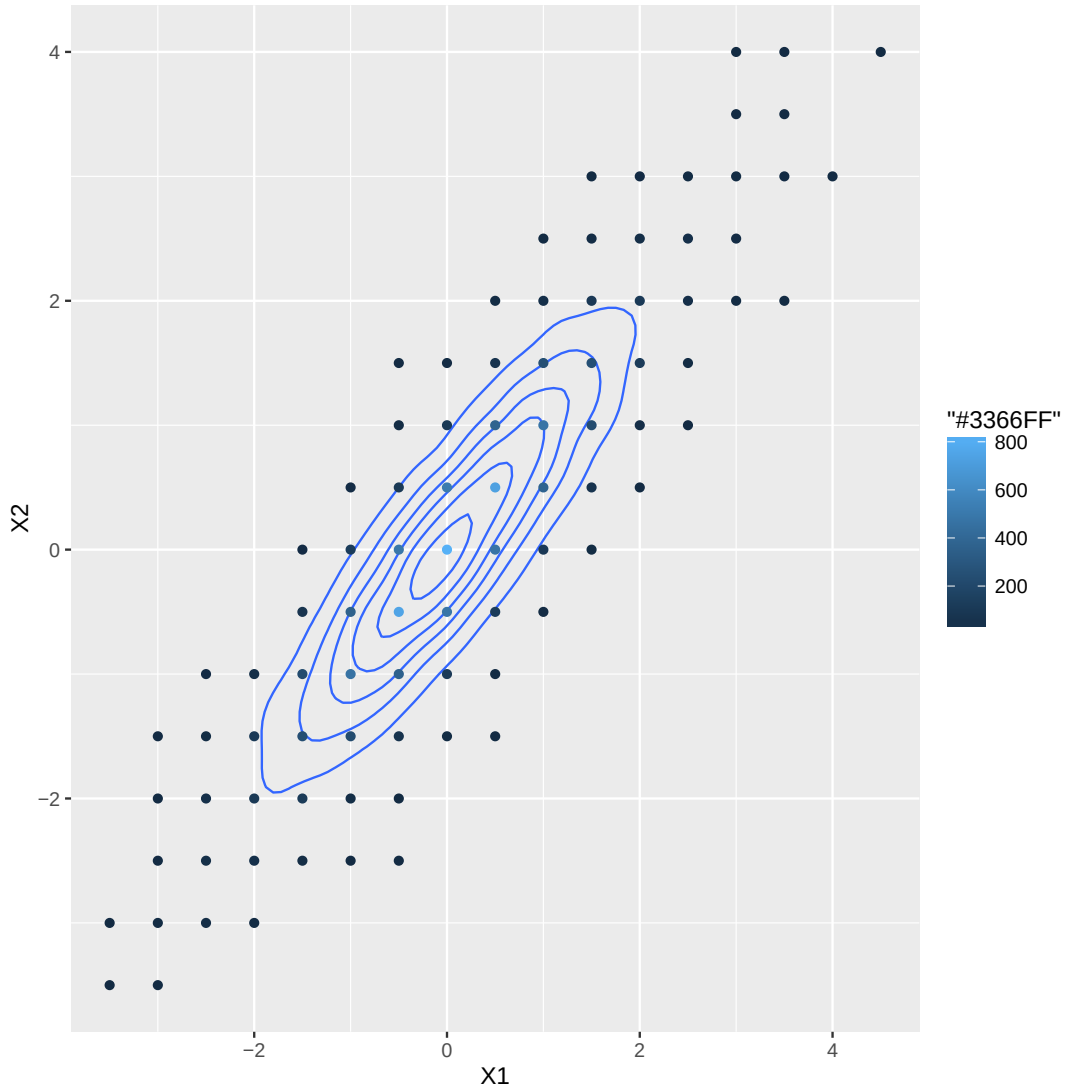

```
## Density family bivariate-D
n_sim <- 10000
discrete_sample <- sim_from_pdtmvn_mixt(n = n_sim, sim_family = "bivariate-D-discretized")
    as.data.frame()

## Error in get_dist_component_params_for_sim_family(sim_family): Invalid
sim_family
```

```
continuous_sample <- sim_from_pdtmvn_mixt(n = n_sim, sim_family = "bivariate-D") %>%
    as.data.frame()

## Error in get_dist_component_params_for_sim_family(sim_family): Invalid
sim_family

discrete_sample_counts <- discrete_sample %>%
    count(X1, X2)

pd <- ggplot() +
    geom_density_2d(aes(x = X1, y = X2), data = continuous_sample) +
    geom_point(aes(x = X1, y = X2, colour = n), data = discrete_sample_counts)
pd
```




```
## Density family multivariate-2d
n_sim <- 10000
discrete_sample <- sim_from_pdtmvn_mixt(n = n_sim, sim_family = "multivariate-2d-discretiz
    as.data.frame()
continuous_sample <- sim_from_pdtmvn_mixt(n = n_sim, sim_family = "multivariate-2d") %>%
    as.data.frame()
discrete_sample_counts <- discrete_sample %>%
```

```
    count(X1, X2)

pd <- ggplot() +
    geom_density_2d(aes(x = X1, y = X2), data = continuous_sample) +
    geom_point(aes(x = X1, y = X2, colour = n), data = discrete_sample_counts)
pd
```



Examples

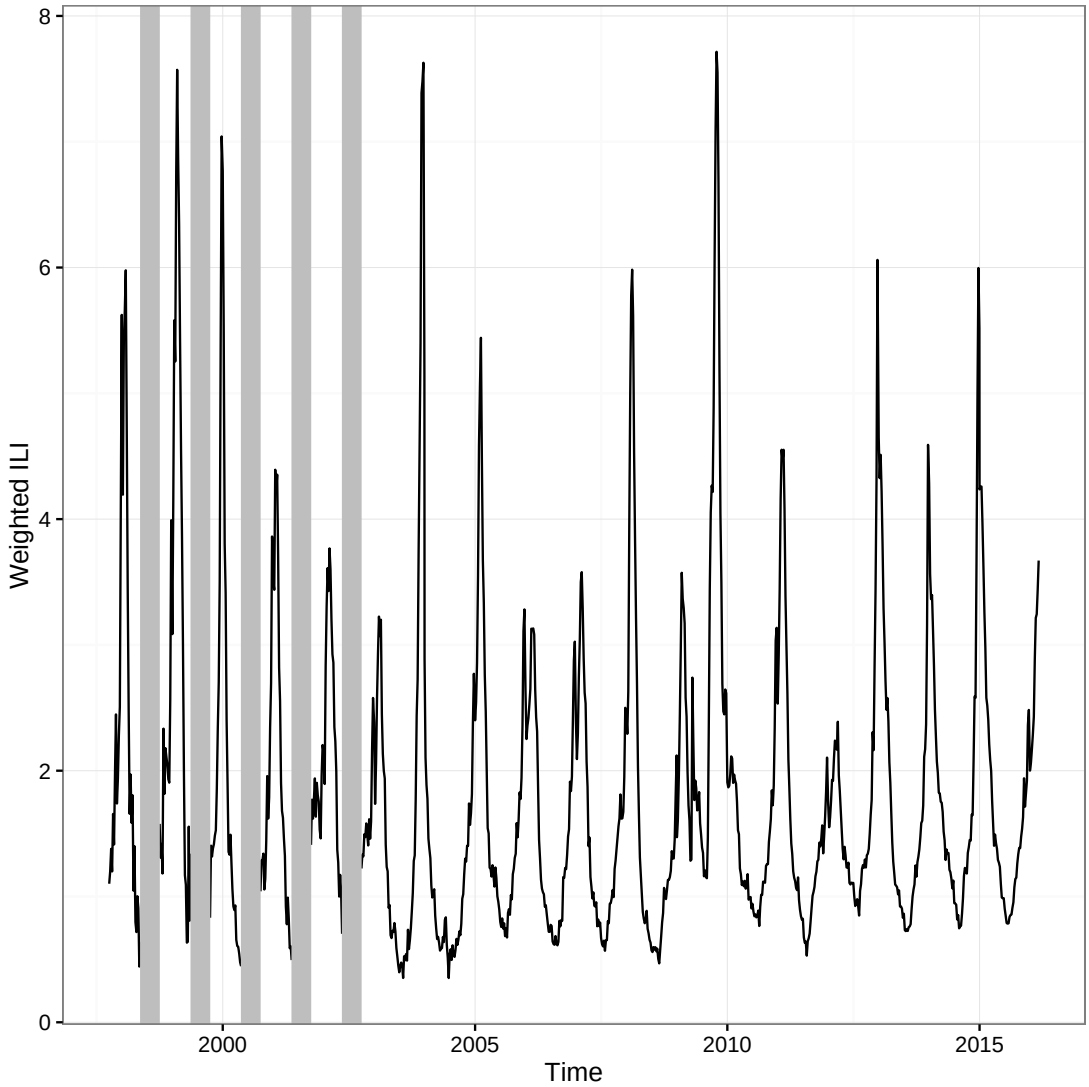
In this Section, we illustrate the methods through applications to prediction in examples with several real time series data sets.

Example 1: Influenza Prediction

In our first and simplest example, we apply the method for prediction of influenza with prediction horizons of 1 through 4 weeks. Data on influenza incidence are available through R's `cdcfluview` package. Here we create a data set with a nationally aggregated measure of flu incidence

```
##
  |
  |                                                      |   0%
  |
  |++++++++++++++++++++++++++++++++++++++++++++++++++++++| 100%

## Warning in eval(substitute(expr), envir, enclos): NAs introduced by coercion

## 'data.frame': 963 obs. of  7 variables:
##  $ region.type : chr  "National" "National" "National" "National" ...
##  $ region      : chr  "X" "X" "X" "X" ...
##  $ year        : int  1997 1997 1997 1997 1997 1997 1997 1997 1997 1997 ...
##  $ week        : int  40 41 42 43 44 45 46 47 48 49 ...
##  $ weighted_ili: num  1.1 1.2 1.38 1.2 1.66 ...
##  $ time        : POSIXct, format: "1997-10-01" "1997-10-08" ...
##  $ time_index  : int  1 2 3 4 5 6 7 8 9 10 ...
```

We plot the `total_cases` measure over time, representing missing values with vertical grey lines. The low season was not measured in the first few years.



There are several methods that we could employ to handle these missing data:

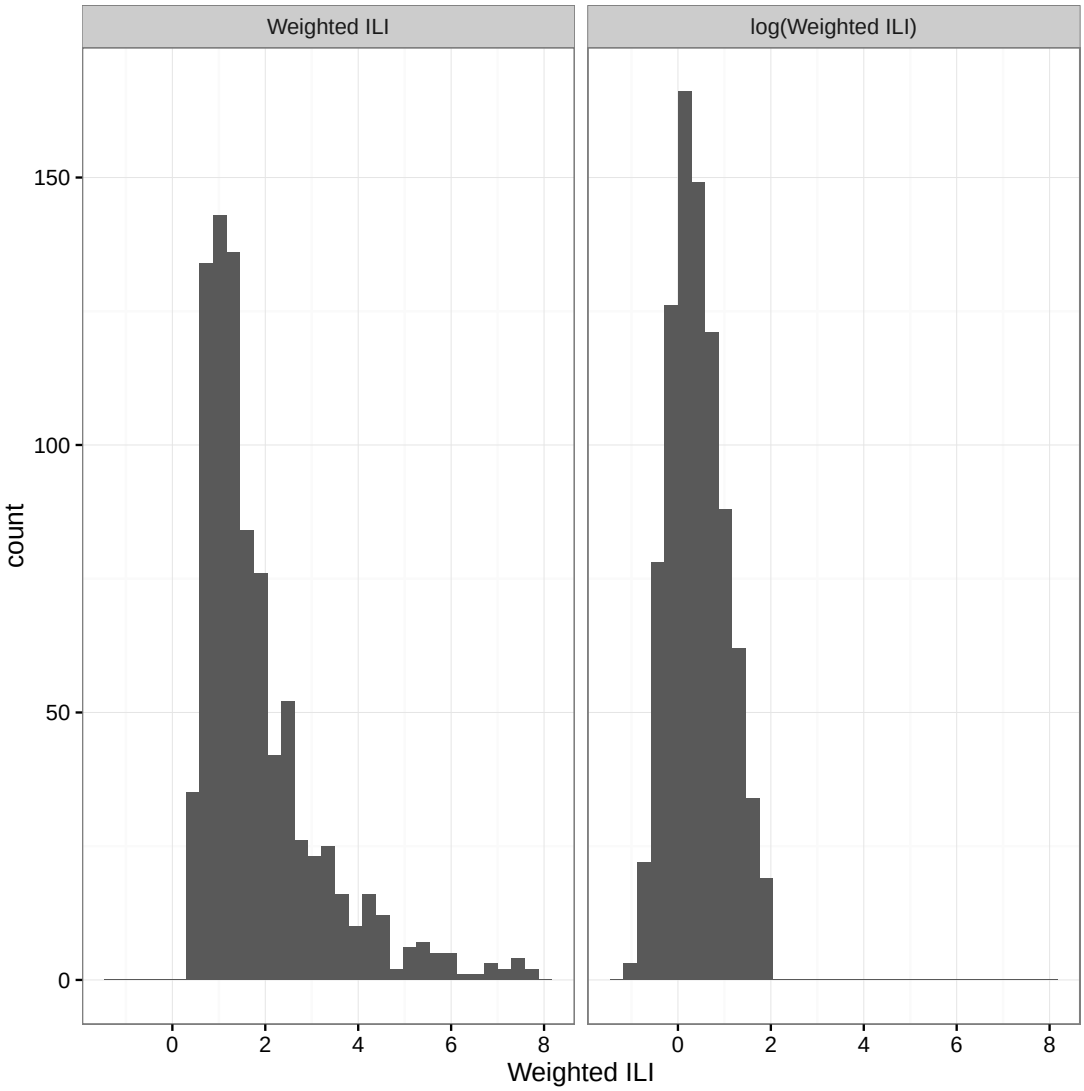
1. Impute the missing values. They are all in the low season, so this should be relatively easy to do.
2. Drop all data up through the last NA.
3. Use the data that are available.

Of these approaches, the first is probably preferred. The concern with the second is that we are not making use of all of the available data. The potential concern with the third is that in the

data used in estimation, there will be more examples of prediction of values in the high season using values in the high season and middle of the season than of prediction of values in the high season using values in the low season. This could potentially affect our inference. However, we do not expect this effect to be large, so we proceed with this option for the purposes of this example.

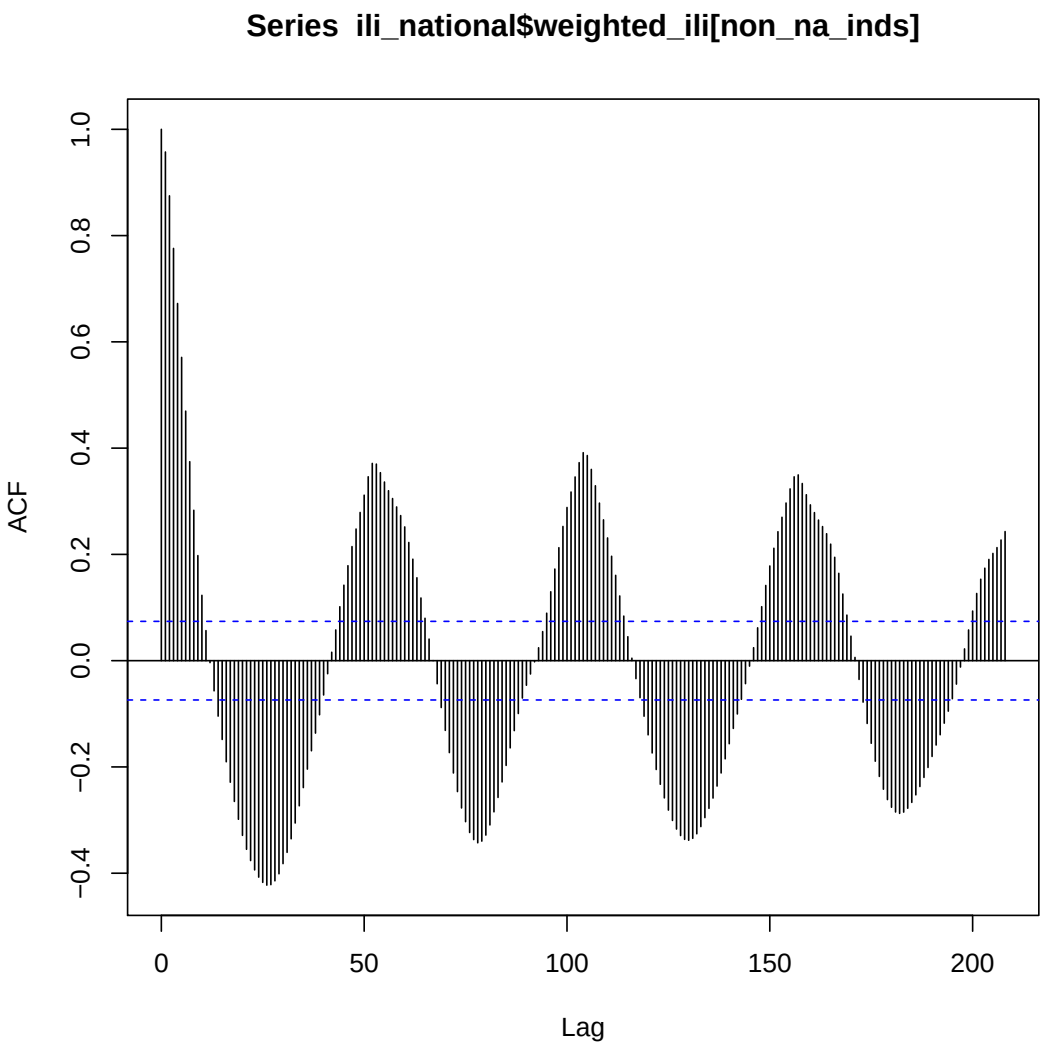
We also plot histograms of the observed total cases on the original scale and on the log scale.

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 190 rows containing non-finite values (stat_bin).
```



These plots demonstrate that total cases follows an approximately log-normal distribution. In the application below, we will consider modeling these data on both the original scale and the log scale. Intuitively, since we are using a kernel that is obtained from a Gaussian, modeling the data on the log scale should yield better performance. On the other hand, the performance gain may be negligible if we have enough data.

Finally, we plot the autocorrelation function:

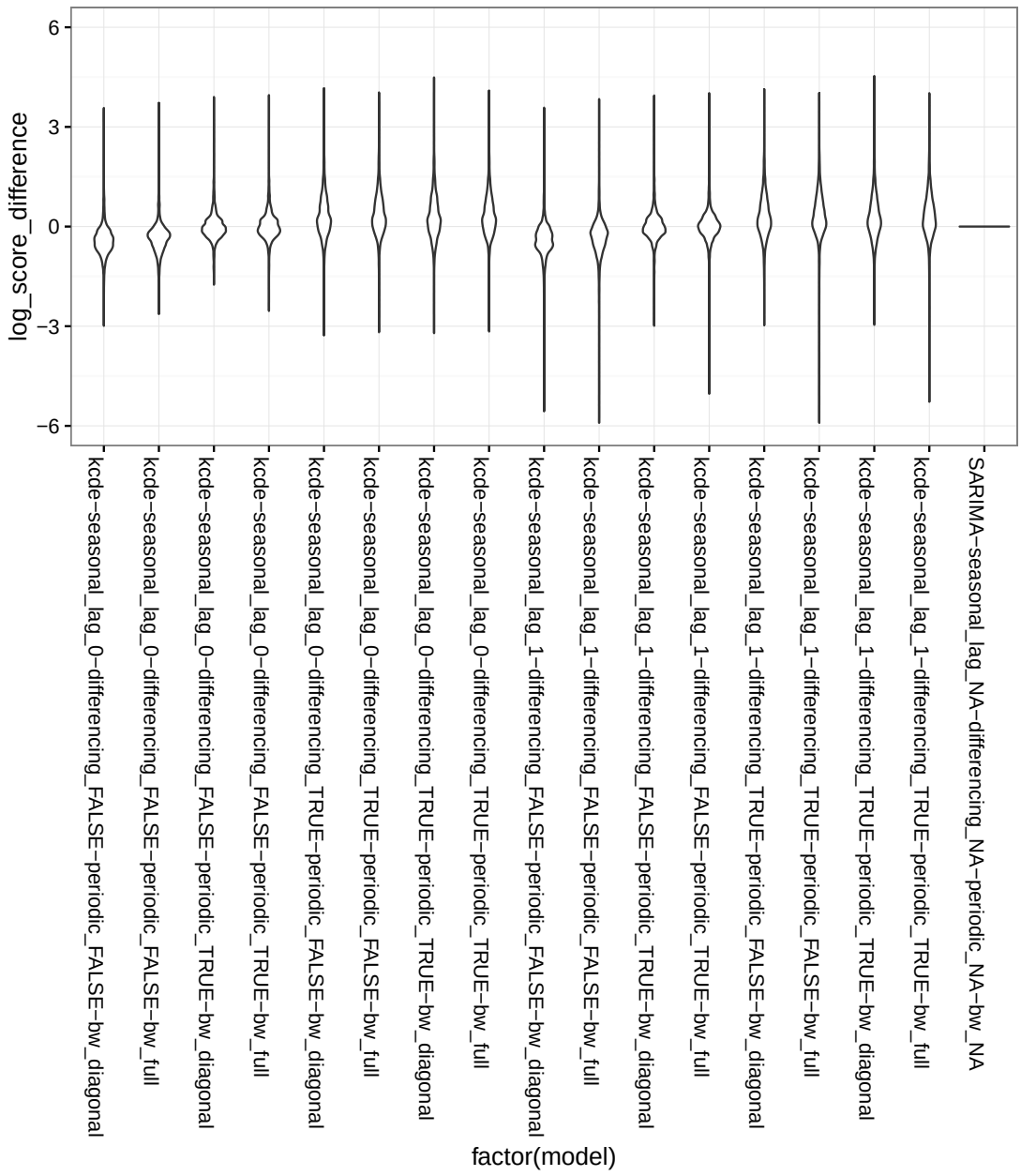


This plot illustrates the annual periodicity that was also visible in the initial data plot above. There is no apparent evidence of longer term annual cycles. We therefore include a periodic kernel acting on the time index with a period of 52.2 weeks (the length of the period is motivated by the fact that in our data, there is a year with 53 weeks once every 5 or 6 years).

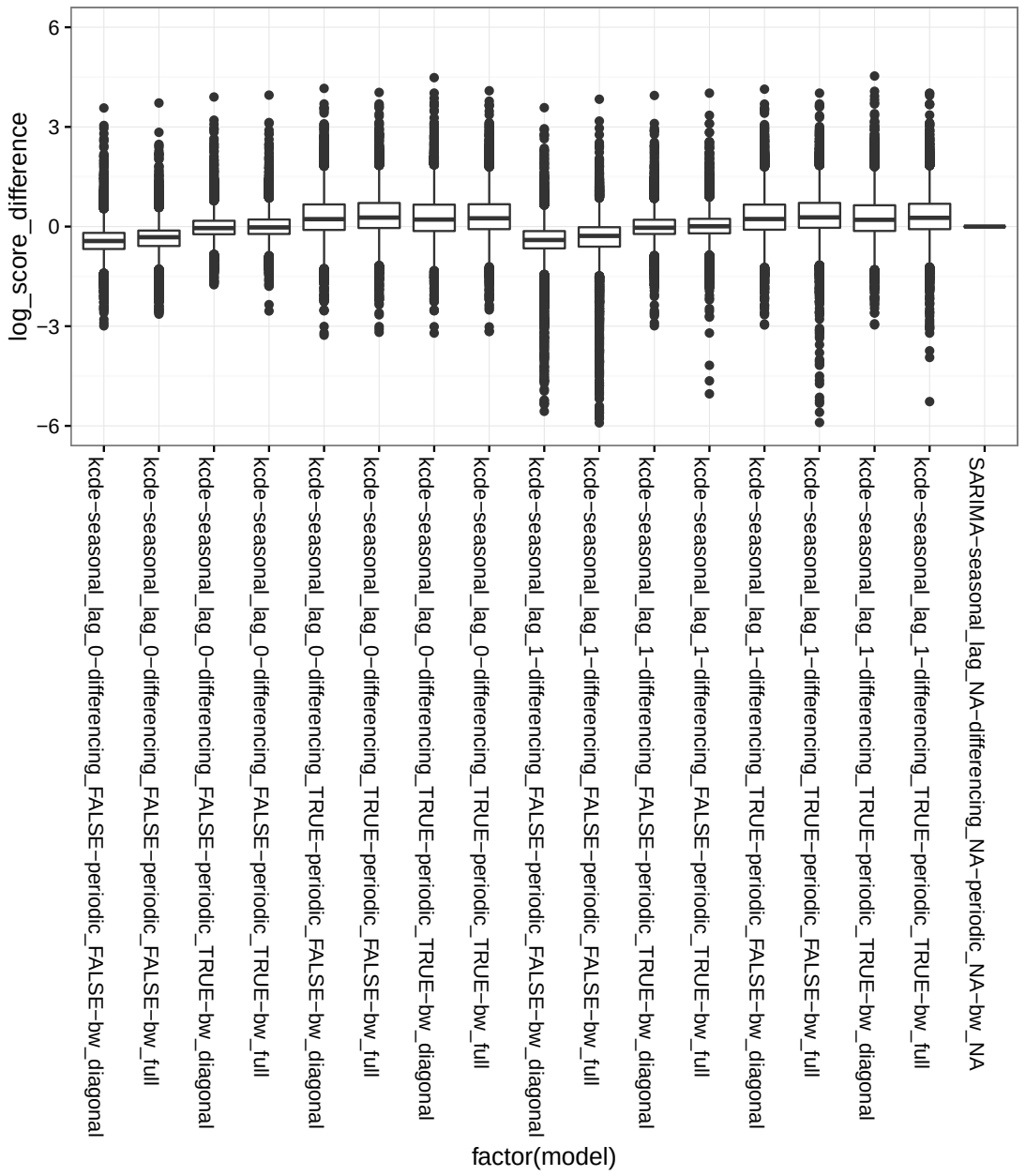
We now do some set up for estimation and prediction with kcde. First, we create a list with parameters that specify the kernel function components.

log_score_difference
0
−10
−20
−30
−40
−50
factor(model)
SARIMA−seasonal_lag_NA−differencing_NA−periodic_NA−bw_NA
kcde−seasonal_lag_1−differencing_TRUE−periodic_TRUE−bw_full
kcde−seasonal_lag_1−differencing_TRUE−periodic_TRUE−bw_diagonal
kcde−seasonal_lag_1−differencing_TRUE−periodic_FALSE−bw_full
kcde−seasonal_lag_1−differencing_TRUE−periodic_FALSE−bw_diagonal
kcde−seasonal_lag_1−differencing_FALSE−periodic_TRUE−bw_full
kcde−seasonal_lag_1−differencing_FALSE−periodic_TRUE−bw_diagonal
kcde−seasonal_lag_1−differencing_FALSE−periodic_FALSE−bw_full
kcde−seasonal_lag_1−differencing_FALSE−periodic_FALSE−bw_diagonal
kcde−seasonal_lag_0−differencing_TRUE−periodic_TRUE−bw_full
kcde−seasonal_lag_0−differencing_TRUE−periodic_TRUE−bw_diagonal
kcde−seasonal_lag_0−differencing_TRUE−periodic_FALSE−bw_full
kcde−seasonal_lag_0−differencing_TRUE−periodic_FALSE−bw_diagonal
kcde−seasonal_lag_0−differencing_FALSE−periodic_TRUE−bw_full
kcde−seasonal_lag_0−differencing_FALSE−periodic_TRUE−bw_diagonal
kcde−seasonal_lag_0−differencing_FALSE−periodic_FALSE−bw_full
kcde−seasonal_lag_0−differencing_FALSE−periodic_FALSE−bw_diagonal

```
## Warning: Removed 58 rows containing non-finite values (stat_ydensity).
```

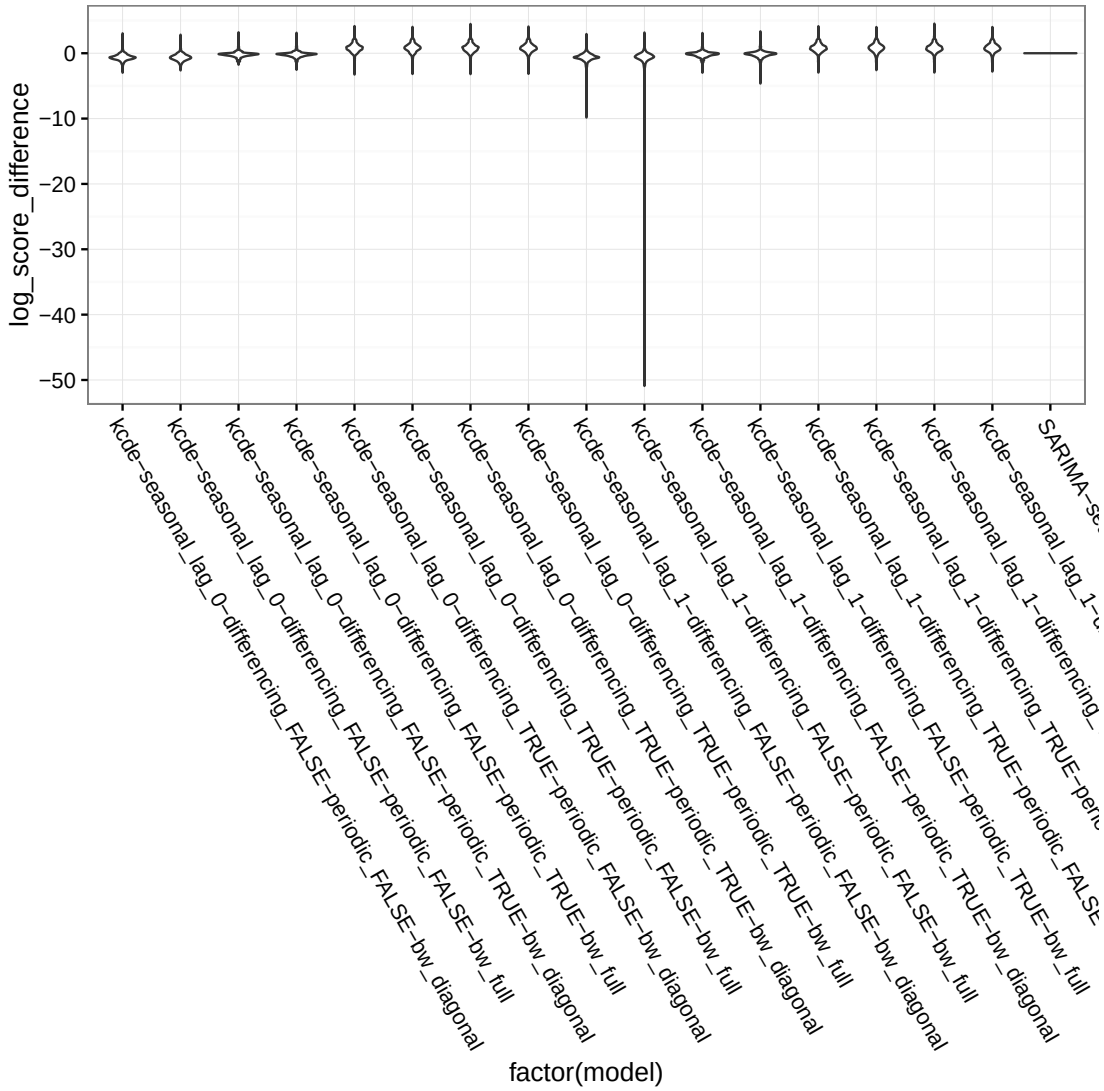


```
## Warning: Removed 58 rows containing non-finite values (stat_boxplot).
```



```
## Warning: Removed 58 rows containing missing values (geom_point).
```

log_score_difference
6
3
0
−3
−6
factor(model)
SARIMA−seasonal_lag_NA−differencing_NA−periodic_NA−bw_NA
kcde−seasonal_lag_1−differencing_TRUE−periodic_TRUE−bw_full
kcde−seasonal_lag_1−differencing_TRUE−periodic_TRUE−bw_diagonal
kcde−seasonal_lag_1−differencing_TRUE−periodic_FALSE−bw_full
kcde−seasonal_lag_1−differencing_TRUE−periodic_FALSE−bw_diagonal
kcde−seasonal_lag_1−differencing_FALSE−periodic_TRUE−bw_full
kcde−seasonal_lag_1−differencing_FALSE−periodic_TRUE−bw_diagonal
kcde−seasonal_lag_1−differencing_FALSE−periodic_FALSE−bw_full
kcde−seasonal_lag_1−differencing_FALSE−periodic_FALSE−bw_diagonal
kcde−seasonal_lag_0−differencing_TRUE−periodic_TRUE−bw_full
kcde−seasonal_lag_0−differencing_TRUE−periodic_TRUE−bw_diagonal
kcde−seasonal_lag_0−differencing_TRUE−periodic_FALSE−bw_full
kcde−seasonal_lag_0−differencing_TRUE−periodic_FALSE−bw_diagonal
kcde−seasonal_lag_0−differencing_FALSE−periodic_TRUE−bw_full
kcde−seasonal_lag_0−differencing_FALSE−periodic_TRUE−bw_diagonal
kcde−seasonal_lag_0−differencing_FALSE−periodic_FALSE−bw_full
kcde−seasonal_lag_0−differencing_FALSE−periodic_FALSE−bw_diagonal



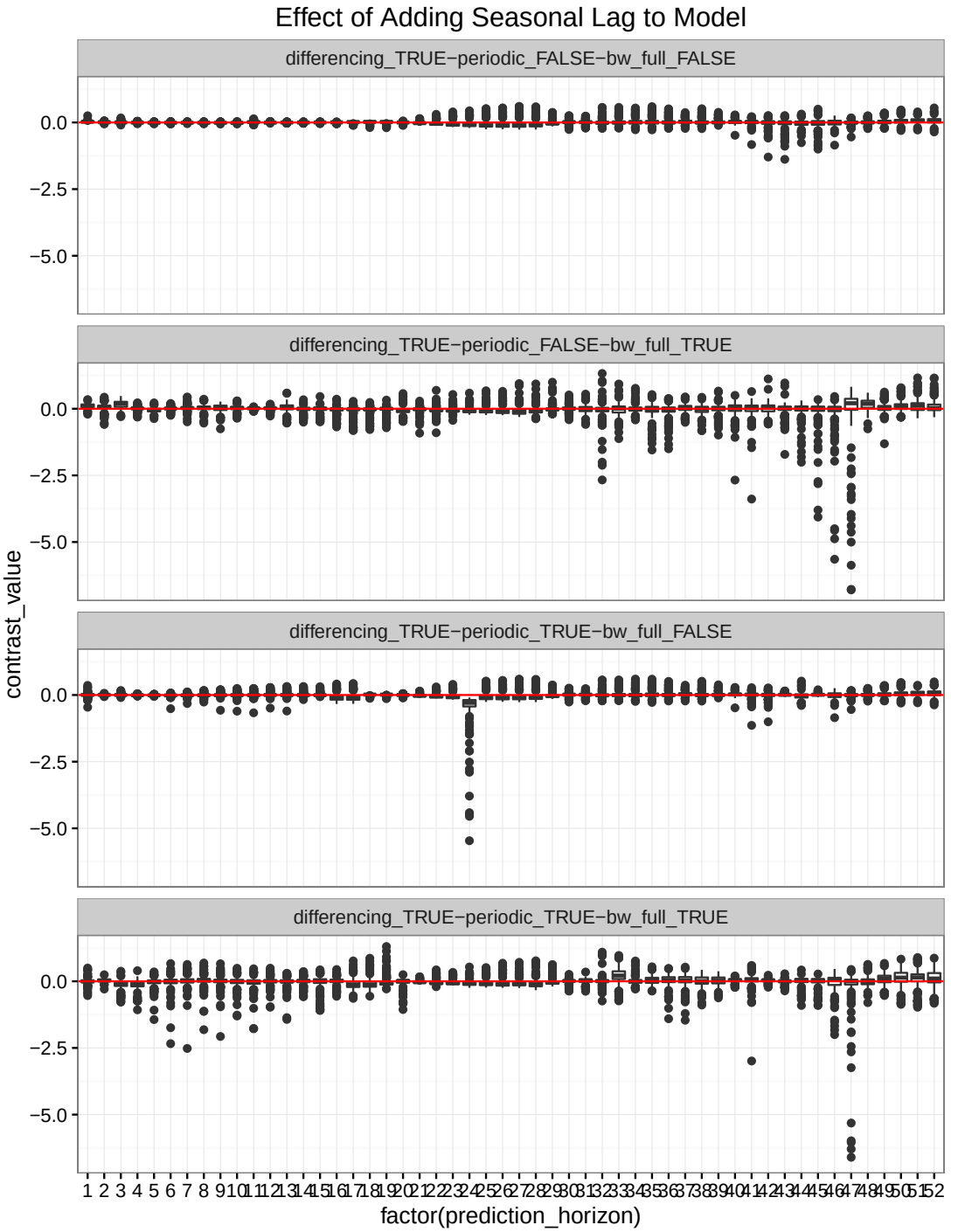

```
## Error in fortify(data): object 'ili_prediction_log_score_diffs' not found
```

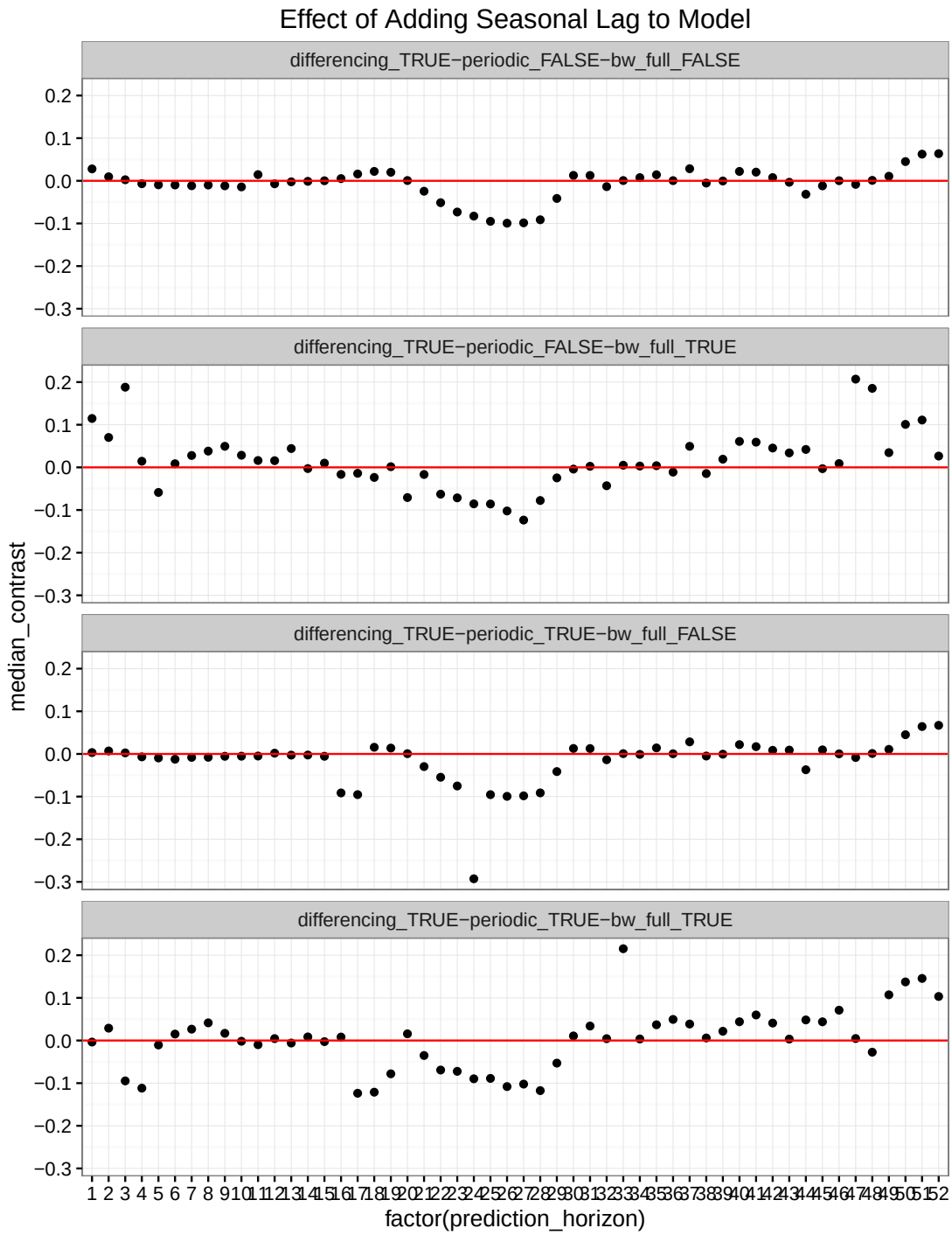
```
## Error: Duplicate identifiers for rows (1, 97813), (2, 97814), (3, 97815),
(4, 97816), (5, 97817), (6, 97818), (7, 97819), (8, 97820), (9, 97821), (10,
97822), (11, 97823), (12, 97824), (13, 97825), (14, 97826), (15, 97827), (16,
97828), (17, 97829), (18, 97830), (19, 97831), (20, 97832), (21, 97833), (22,
```

```
97834), (23, 97835), (24, 97836), (25, 97837), (26, 97838), (27, 97839), (28,
97840), (29, 97841), (30, 97842), (31, 97843), (32, 97844), (33, 97845), (34,
97846), (35, 97847), (36, 97848), (37, 97849), (38, 97850), (39, 97851), (40,
97852), (41, 97853), (42, 97854), (43, 97855), (44, 97856), (45, 97857), (46,
97858), (47, 97859), (48, 97860), (49, 97861), (50, 97862), (51, 97863), (52,
97864), (53, 97865), (54, 97866), (55, 97867), (56, 97868), (57, 97869), (58,
97870), (59, 97871), (60, 97872), (61, 97873), (62, 97874), (63, 97875), (64,
97876), (65, 97877), (66, 97878), (67, 97879), (68, 97880), (69, 97881), (70,
97882), (71, 97883), (72, 97884), (73, 97885), (74, 97886), (75, 97887), (76,
97888), (77, 97889), (78, 97890), (79, 97891), (80, 97892), (81, 97893), (82,
97894), (83, 97895), (84, 97896), (85, 97897), (86, 97898), (87, 97899), (88,
97900), (89, 97901), (90, 97902), (91, 97903), (92, 97904), (93, 97905), (94,
97906), (95, 97907), (96, 97908), (97, 97909), (98, 97910), (99, 97911), (100,
97912), (101, 97913), (102, 97914), (103, 97915), (104, 97916), (105, 97917),
(106, 97918), (107, 97919), (108, 97920), (109, 97921), (110, 97922), (111,
97923), (112, 97924), (113, 97925), (114, 97926), (115, 97927), (116, 97928),
(117, 97929), (118, 97930), (119, 97931), (120, 97932), (121, 97933), (122,
97934), (123, 97935), (124, 97936), (125, 97937), (126, 97938), (127, 97939),
(128, 97940), (129, 97941), (130, 97942), (131, 97943), (132, 97944), (133,
97945), (134, 97946), (135, 97947), (136, 97948), (137, 97949), (138, 97950),
(139, 97951), (140, 97952), (141, 97953), (142, 97954), (143, 97955), (144,
97956), (145, 97957), (146, 97958), (147, 97959), (148, 97960), (149, 97961),
(150, 97962), (151, 97963), (152, 97964), (153, 97965), (154, 97966), (155,
97967), (156, 97968), (157, 97969), (158, 97970), (159, 97971), (160, 97972),
(161, 97973), (162, 97974), (163, 97975), (164, 97976), (165, 97977), (166,
97978), (167, 97979), (168, 97980), (169, 97981), (170, 97982), (171, 97983),
(172, 97984), (173, 97985), (174, 97986), (175, 97987), (176, 97988), (177,
97989), (178, 97990), (179, 97991), (180, 97992), (181, 97993), (182, 97994),
(183, 97995), (184, 97996), (185, 97997), (186, 97998), (187, 97999), (188,
98000), (189, 98001), (190, 98002), (191, 98003), (192, 98004), (193, 98005),
(194, 98006), (195, 98007), (196, 98008), (197, 98009), (198, 98010), (199,
98011), (200, 98012), (201, 98013), (202, 98014), (203, 98015), (204, 98016),
(205, 98017), (206, 98018), (207, 98019), (208, 98020), (209, 98021), (210,
98022), (211, 98023), (212, 98024), (213, 98025), (214, 98026), (215, 98027),
(216, 98028), (217, 98029), (218, 98030), (219, 98031), (220, 98032), (221,
98033), (222, 98034), (223, 98035), (224, 98036), (225, 98037), (226, 98038),
(227, 98039), (228, 98040), (229, 98041), (230, 98042), (231, 98043), (232,
98044), (233, 98045), (234, 98046), (235, 98047), (236, 98048), (237, 98049),
(238, 98050), (239, 98051), (240, 98052), (241, 98053), (242, 98054), (243,
98055), (244, 98056), (245, 98057), (246, 98058), (247, 98059), (248, 98060),
(249, 98061), (250, 98062), (251, 98063), (252, 98064), (253, 98065), (254,
98066), (255, 98067), (256, 98068), (257, 98069), (258, 98070), (259, 98071),
```

```
(260, 98072), (261, 98073), (262, 98074), (263, 98075), (264, 98076), (265,
98077), (266, 98078), (267, 98079), (268, 98080), (269, 98081), (270, 98082),
(271, 98083), (272, 98084), (273, 98085), (274, 98086), (275, 98087), (276,
98088), (277, 98089), (278, 98090), (279, 98091), (280, 98092), (281, 98093),
(282, 98094), (283, 98095), (284, 98096), (285, 98097), (286, 98098), (287,
98099), (288, 98100), (289, 98101), (290, 98102), (291, 98103), (292, 98104),
(293, 98105), (294, 98106), (295, 98107), (296, 98108), (297, 98109), (298,
98110), (299, 98111), (300, 98112), (301, 98113), (302, 98114), (303, 98115),
(304, 98116), (305, 98117), (306, 98118), (307, 98119), (308, 98120), (309,
98121), (310, 98122), (311, 98123), (312, 98124), (313, 98125), (314, 98126),
(315, 98127), (316, 98128), (317, 98129), (318, 98130), (319, 98131), (320,
98132), (321, 98133), (322, 98134), (323, 98135), (324, 98136), (325, 98137),
(326, 98138), (327, 98139), (328, 98140), (329, 98141), (330, 98142), (331,
98143), (332, 98144), (333, 98145), (334, 98146), (335, 98147), (336, 98148),
(337, 98149), (338, 98150), (339, 98151), (340, 98152), (341, 98153), (342,
98154), (343, 98155), (344, 98156), (345, 98157), (346, 98158), (347, 98159),
(348, 98160), (349, 98161), (350, 98162), (351, 98163), (352, 98164), (353,
98165), (354, 98166), (355, 98167), (356, 98168), (357, 98169), (358, 98170),
(359, 98171), (360, 98172), (361, 98173), (362, 98174), (363, 98175), (364,
98176), (365, 98177), (366, 98178), (367, 98179), (368, 98180), (369, 98181),
(370, 98182), (371, 98183), (372, 98184), (373, 98185), (374, 98186), (375,
98187), (376, 98188), (377, 98189), (378, 98190), (379, 98191), (380, 98192),
(381, 98193), (382, 98194), (383, 98195), (384, 98196), (385, 98197), (386,
98198), (387, 98199), (388, 98200), (389, 98201), (390, 98202), (391, 98203),
(392, 98204), (393, 98205), (394, 98206), (395, 98207), (396, 98208), (397,
98209), (398, 98210), (399, 98211), (400, 98212), (401, 98213), (402, 98214),
(403, 98215), (404, 98216), (405, 98217), (406, 98218), (407, 98219), (408,
98220), (409, 98221), (410, 98222), (411, 98223), (412, 98224), (413, 98225),
(414, 98226), (415, 98227), (416, 98228), (417, 98229), (418, 98230), (419,
98231), (420, 98232), (421, 98233), (422, 98234), (423, 98235), (424, 98236),
(425, 98237), (426, 98238), (427, 98239), (428, 98240), (429, 98241), (430,
98242), (431, 98243), (432, 98244), (433, 98245), (434, 98246), (435, 98247),
(436, 98248), (437, 98249), (438, 98250), (439, 98251), (440, 98252), (441,
98253), (442, 98254), (443, 98255), (444, 98256), (445, 98257), (446, 98258),
(447, 98259), (448, 98260), (449, 98261), (450, 98262), (451, 98263), (452,
98264), (453, 98265), (454, 98266), (455, 98267), (456, 98268), (457, 98269),
(458, 98270), (459, 98271), (460, 98272), (461, 98273), (462, 98274), (463,
98275), (464, 98276), (465, 98277), (466, 98278), (467, 98279), (468, 98280),
(469, 98281), (470, 98282), (471, 98283), (472, 98284), (473, 98285), (474,
98286), (475, 98287), (476, 98288), (477, 98289), (478, 98290), (479, 98291),
(480, 98292), (481, 98293), (482, 98294), (483, 98295), (484, 98296), (485,
98297), (486, 98298), (487, 98299), (488, 98300), (489, 98301), (490, 98302),
```

```
(491, 98303), (492, 98304), (493, 98305), (494, 98306), (495, 98307), (496,
98308), (497, 98309), (498, 98310), (499, 98311), (500, 98312), (501, 98313),
(502, 98314), (503, 98315), (504, 98316), (505, 98317), (506, 98318), (507,
98319), (508, 98320), (509, 98321), (510, 98322), (511, 98323), (512, 98324),
(513, 98325), (514, 98326), (515, 98327), (516, 98328), (517, 98329), (518,
98330), (519, 98331), (520, 98332), (521, 98333), (522, 98334), (523, 98335),
(524, 98336), (525, 98337), (526, 98338), (527, 98339), (528, 98340), (529,
98341), (530, 98342), (531, 98343), (532, 98344), (533, 98345), (534, 98346),
(535, 98347), (536, 98348), (537, 98349), (538, 98350), (539, 98351), (540,
98352), (541, 98353), (542, 98354), (543, 98355), (544, 98356), (545, 98357),
(546, 98358), (547, 98359), (548, 98360), (549, 98361), (550, 98362), (551,
98363), (552, 98364), (553, 98365), (554, 98366), (555, 98367), (556, 98368),
(557, 98369), (558, 98370), (559, 98371), (560, 98372), (561, 98373), (562,
98374), (563, 98375), (564, 98376), (565, 98377), (566, 98378), (567, 98379),
(568, 98380), (569, 98381), (570, 98382), (571, 98383), (572, 98384), (573,
98385), (574, 98386), (575, 98387), (576, 98388), (577, 98389), (578, 98390),
(579, 98391), (580, 98392), (581, 98393), (582, 98394), (583, 98395), (584,
98396), (585, 98397), (586, 98398), (587, 98399), (588, 98400), (589, 98401),
(590, 98402), (591, 9
```


Effect of Adding Seasonal Lag to Model
differencing_TRUE−periodic_FALSE−bw_full_FALSE
differencing_TRUE−periodic_FALSE−bw_full_TRUE
differencing_TRUE−periodic_TRUE−bw_full_FALSE
differencing_TRUE−periodic_TRUE−bw_full_TRUE
contrast_value
0.0
−2.5
−5.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
factor(prediction_horizon)


Effect of Adding Seasonal Lag to Model
differencing_TRUE−periodic_FALSE−bw_full_FALSE
differencing_TRUE−periodic_FALSE−bw_full_TRUE
differencing_TRUE−periodic_TRUE−bw_full_FALSE
differencing_TRUE−periodic_TRUE−bw_full_TRUE
median_contrast
0.2
0.1
0.0
−0.1
−0.2
−0.3
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
factor(prediction_horizon)

```
## Error: Duplicate identifiers for rows (1, 97813), (2, 97814), (3, 97815),
(4, 97816), (5, 97817), (6, 97818), (7, 97819), (8, 97820), (9, 97821), (10,
97822), (11, 97823), (12, 97824), (13, 97825), (14, 97826), (15, 97827), (16,
97828), (17, 97829), (18, 97830), (19, 97831), (20, 97832), (21, 97833), (22,
97834), (23, 97835), (24, 97836), (25, 97837), (26, 97838), (27, 97839), (28,
97840), (29, 97841), (30, 97842), (31, 97843), (32, 97844), (33, 97845), (34,
97846), (35, 97847), (36, 97848), (37, 97849), (38, 97850), (39, 97851), (40,
97852), (41, 97853), (42, 97854), (43, 97855), (44, 97856), (45, 97857), (46,
97858), (47, 97859), (48, 97860), (49, 97861), (50, 97862), (51, 97863), (52,
97864), (53, 97865), (54, 97866), (55, 97867), (56, 97868), (57, 97869), (58,
97870), (59, 97871), (60, 97872), (61, 97873), (62, 97874), (63, 97875), (64,
97876), (65, 97877), (66, 97878), (67, 97879), (68, 97880), (69, 97881), (70,
97882), (71, 97883), (72, 97884), (73, 97885), (74, 97886), (75, 97887), (76,
97888), (77, 97889), (78, 97890), (79, 97891), (80, 97892), (81, 97893), (82,
97894), (83, 97895), (84, 97896), (85, 97897), (86, 97898), (87, 97899), (88,
97900), (89, 97901), (90, 97902), (91, 97903), (92, 97904), (93, 97905), (94,
97906), (95, 97907), (96, 97908), (97, 97909), (98, 97910), (99, 97911), (100,
97912), (101, 97913), (102, 97914), (103, 97915), (104, 97916), (105, 97917),
(106, 97918), (107, 97919), (108, 97920), (109, 97921), (110, 97922), (111,
97923), (112, 97924), (113, 97925), (114, 97926), (115, 97927), (116, 97928),
(117, 97929), (118, 97930), (119, 97931), (120, 97932), (121, 97933), (122,
97934), (123, 97935), (124, 97936), (125, 97937), (126, 97938), (127, 97939),
(128, 97940), (129, 97941), (130, 97942), (131, 97943), (132, 97944), (133,
97945), (134, 97946), (135, 97947), (136, 97948), (137, 97949), (138, 97950),
(139, 97951), (140, 97952), (141, 97953), (142, 97954), (143, 97955), (144,
97956), (145, 97957), (146, 97958), (147, 97959), (148, 97960), (149, 97961),
(150, 97962), (151, 97963), (152, 97964), (153, 97965), (154, 97966), (155,
97967), (156, 97968), (157, 97969), (158, 97970), (159, 97971), (160, 97972),
(161, 97973), (162, 97974), (163, 97975), (164, 97976), (165, 97977), (166,
97978), (167, 97979), (168, 97980), (169, 97981), (170, 97982), (171, 97983),
(172, 97984), (173, 97985), (174, 97986), (175, 97987), (176, 97988), (177,
97989), (178, 97990), (179, 97991), (180, 97992), (181, 97993), (182, 97994),
(183, 97995), (184, 97996), (185, 97997), (186, 97998), (187, 97999), (188,
98000), (189, 98001), (190, 98002), (191, 98003), (192, 98004), (193, 98005),
(194, 98006), (195, 98007), (196, 98008), (197, 98009), (198, 98010), (199,
98011), (200, 98012), (201, 98013), (202, 98014), (203, 98015), (204, 98016),
(205, 98017), (206, 98018), (207, 98019), (208, 98020), (209, 98021), (210,
98022), (211, 98023), (212, 98024), (213, 98025), (214, 98026), (215, 98027),
(216, 98028), (217, 98029), (218, 98030), (219, 98031), (220, 98032), (221,
98033), (222, 98034), (223, 98035), (224, 98036), (225, 98037), (226, 98038),
(227, 98039), (228, 98040), (229, 98041), (230, 98042), (231, 98043), (232,
98044), (233, 98045), (234, 98046), (235, 98047), (236, 98048), (237, 98049),
```

```
(238, 98050), (239, 98051), (240, 98052), (241, 98053), (242, 98054), (243,
98055), (244, 98056), (245, 98057), (246, 98058), (247, 98059), (248, 98060),
(249, 98061), (250, 98062), (251, 98063), (252, 98064), (253, 98065), (254,
98066), (255, 98067), (256, 98068), (257, 98069), (258, 98070), (259, 98071),
(260, 98072), (261, 98073), (262, 98074), (263, 98075), (264, 98076), (265,
98077), (266, 98078), (267, 98079), (268, 98080), (269, 98081), (270, 98082),
(271, 98083), (272, 98084), (273, 98085), (274, 98086), (275, 98087), (276,
98088), (277, 98089), (278, 98090), (279, 98091), (280, 98092), (281, 98093),
(282, 98094), (283, 98095), (284, 98096), (285, 98097), (286, 98098), (287,
98099), (288, 98100), (289, 98101), (290, 98102), (291, 98103), (292, 98104),
(293, 98105), (294, 98106), (295, 98107), (296, 98108), (297, 98109), (298,
98110), (299, 98111), (300, 98112), (301, 98113), (302, 98114), (303, 98115),
(304, 98116), (305, 98117), (306, 98118), (307, 98119), (308, 98120), (309,
98121), (310, 98122), (311, 98123), (312, 98124), (313, 98125), (314, 98126),
(315, 98127), (316, 98128), (317, 98129), (318, 98130), (319, 98131), (320,
98132), (321, 98133), (322, 98134), (323, 98135), (324, 98136), (325, 98137),
(326, 98138), (327, 98139), (328, 98140), (329, 98141), (330, 98142), (331,
98143), (332, 98144), (333, 98145), (334, 98146), (335, 98147), (336, 98148),
(337, 98149), (338, 98150), (339, 98151), (340, 98152), (341, 98153), (342,
98154), (343, 98155), (344, 98156), (345, 98157), (346, 98158), (347, 98159),
(348, 98160), (349, 98161), (350, 98162), (351, 98163), (352, 98164), (353,
98165), (354, 98166), (355, 98167), (356, 98168), (357, 98169), (358, 98170),
(359, 98171), (360, 98172), (361, 98173), (362, 98174), (363, 98175), (364,
98176), (365, 98177), (366, 98178), (367, 98179), (368, 98180), (369, 98181),
(370, 98182), (371, 98183), (372, 98184), (373, 98185), (374, 98186), (375,
98187), (376, 98188), (377, 98189), (378, 98190), (379, 98191), (380, 98192),
(381, 98193), (382, 98194), (383, 98195), (384, 98196), (385, 98197), (386,
98198), (387, 98199), (388, 98200), (389, 98201), (390, 98202), (391, 98203),
(392, 98204), (393, 98205), (394, 98206), (395, 98207), (396, 98208), (397,
98209), (398, 98210), (399, 98211), (400, 98212), (401, 98213), (402, 98214),
(403, 98215), (404, 98216), (405, 98217), (406, 98218), (407, 98219), (408,
98220), (409, 98221), (410, 98222), (411, 98223), (412, 98224), (413, 98225),
(414, 98226), (415, 98227), (416, 98228), (417, 98229), (418, 98230), (419,
98231), (420, 98232), (421, 98233), (422, 98234), (423, 98235), (424, 98236),
(425, 98237), (426, 98238), (427, 98239), (428, 98240), (429, 98241), (430,
98242), (431, 98243), (432, 98244), (433, 98245), (434, 98246), (435, 98247),
(436, 98248), (437, 98249), (438, 98250), (439, 98251), (440, 98252), (441,
98253), (442, 98254), (443, 98255), (444, 98256), (445, 98257), (446, 98258),
(447, 98259), (448, 98260), (449, 98261), (450, 98262), (451, 98263), (452,
98264), (453, 98265), (454, 98266), (455, 98267), (456, 98268), (457, 98269),
(458, 98270), (459, 98271), (460, 98272), (461, 98273), (462, 98274), (463,
98275), (464, 98276), (465, 98277), (466, 98278), (467, 98279), (468, 98280),
```

```
(469, 98281), (470, 98282), (471, 98283), (472, 98284), (473, 98285), (474,
98286), (475, 98287), (476, 98288), (477, 98289), (478, 98290), (479, 98291),
(480, 98292), (481, 98293), (482, 98294), (483, 98295), (484, 98296), (485,
98297), (486, 98298), (487, 98299), (488, 98300), (489, 98301), (490, 98302),
(491, 98303), (492, 98304), (493, 98305), (494, 98306), (495, 98307), (496,
98308), (497, 98309), (498, 98310), (499, 98311), (500, 98312), (501, 98313),
(502, 98314), (503, 98315), (504, 98316), (505, 98317), (506, 98318), (507,
98319), (508, 98320), (509, 98321), (510, 98322), (511, 98323), (512, 98324),
(513, 98325), (514, 98326), (515, 98327), (516, 98328), (517, 98329), (518,
98330), (519, 98331), (520, 98332), (521, 98333), (522, 98334), (523, 98335),
(524, 98336), (525, 98337), (526, 98338), (527, 98339), (528, 98340), (529,
98341), (530, 98342), (531, 98343), (532, 98344), (533, 98345), (534, 98346),
(535, 98347), (536, 98348), (537, 98349), (538, 98350), (539, 98351), (540,
98352), (541, 98353), (542, 98354), (543, 98355), (544, 98356), (545, 98357),
(546, 98358), (547, 98359), (548, 98360), (549, 98361), (550, 98362), (551,
98363), (552, 98364), (553, 98365), (554, 98366), (555, 98367), (556, 98368),
(557, 98369), (558, 98370), (559, 98371), (560, 98372), (561, 98373), (562,
98374), (563, 98375), (564, 98376), (565, 98377), (566, 98378), (567, 98379),
(568, 98380), (569, 98381), (570, 98382), (571, 98383), (572, 98384), (573,
98385), (574, 98386), (575, 98387), (576, 98388), (577, 98389), (578, 98390),
(579, 98391), (580, 98392), (581, 98393), (582, 98394), (583, 98395), (584,
98396), (585, 98397), (586, 98398), (587, 98399), (588, 98400), (589, 98401),
(590, 98402), (591, 9
## Error in ggplot(ili_contrast_differencing): object 'ili_contrast_differencing'
not found
```

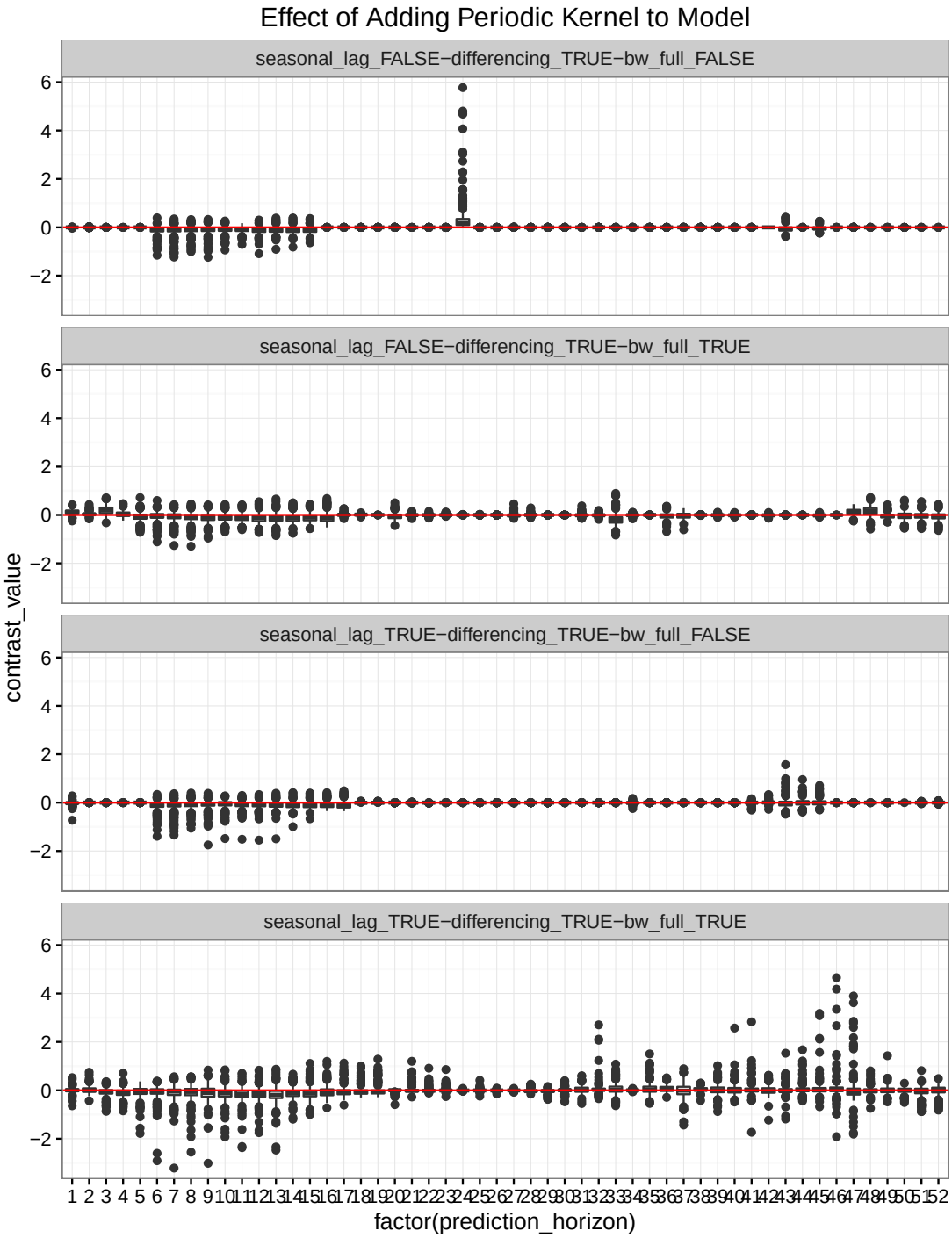
```
## Error in eval(expr, envir, enclos): object 'ili_contrast_differencing' not
found
## Error in ggplot(ili_contrast_differencing_medians): object
'ili_contrast_differencing_medians' not found
```

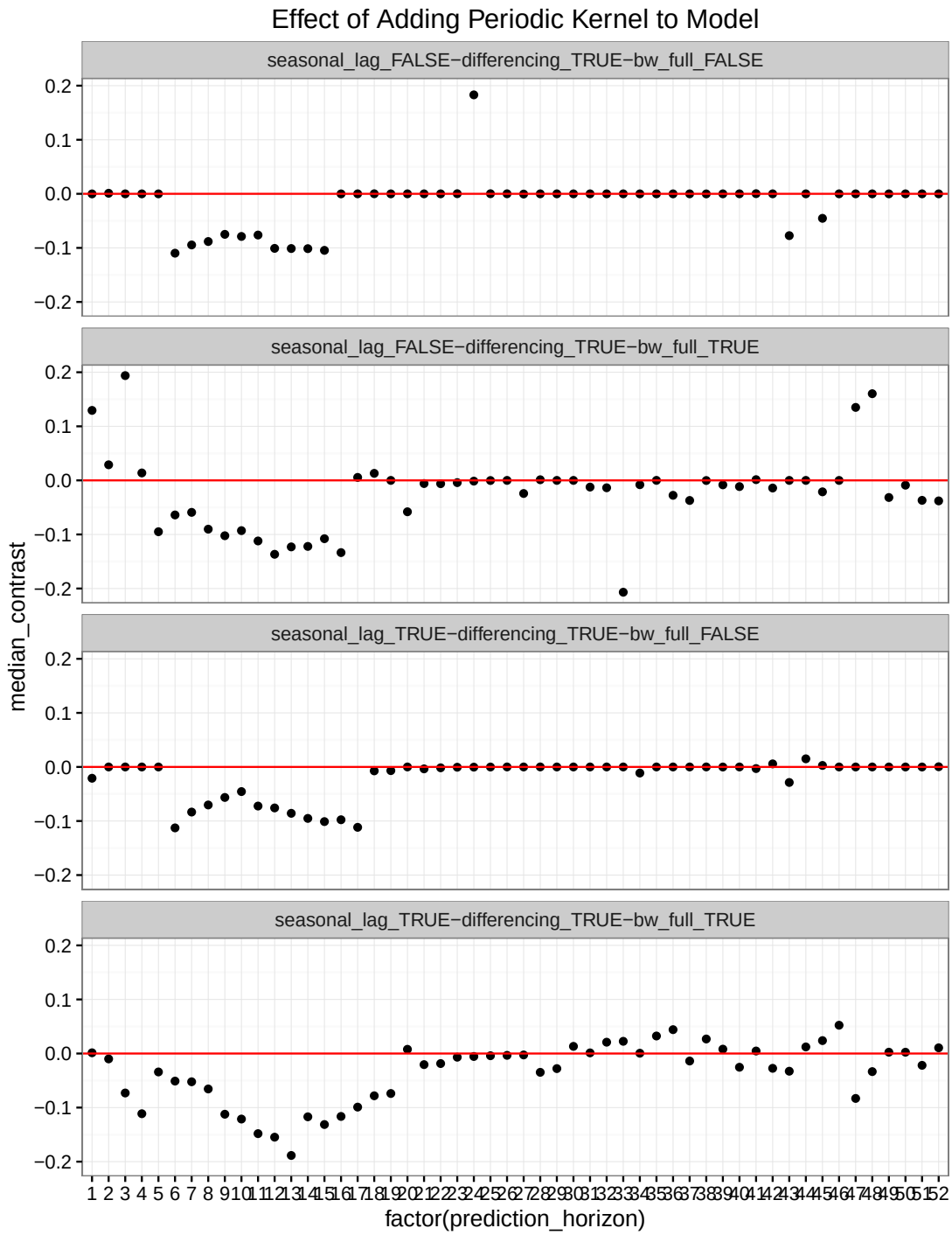
```
## Error: Duplicate identifiers for rows (1, 97813), (2, 97814), (3, 97815),
(4, 97816), (5, 97817), (6, 97818), (7, 97819), (8, 97820), (9, 97821), (10,
97822), (11, 97823), (12, 97824), (13, 97825), (14, 97826), (15, 97827), (16,
97828), (17, 97829), (18, 97830), (19, 97831), (20, 97832), (21, 97833), (22,
97834), (23, 97835), (24, 97836), (25, 97837), (26, 97838), (27, 97839), (28,
97840), (29, 97841), (30, 97842), (31, 97843), (32, 97844), (33, 97845), (34,
97846), (35, 97847), (36, 97848), (37, 97849), (38, 97850), (39, 97851), (40,
97852), (41, 97853), (42, 97854), (43, 97855), (44, 97856), (45, 97857), (46,
97858), (47, 97859), (48, 97860), (49, 97861), (50, 97862), (51, 97863), (52,
97864), (53, 97865), (54, 97866), (55, 97867), (56, 97868), (57, 97869), (58,
```

```
97870), (59, 97871), (60, 97872), (61, 97873), (62, 97874), (63, 97875), (64,
97876), (65, 97877), (66, 97878), (67, 97879), (68, 97880), (69, 97881), (70,
97882), (71, 97883), (72, 97884), (73, 97885), (74, 97886), (75, 97887), (76,
97888), (77, 97889), (78, 97890), (79, 97891), (80, 97892), (81, 97893), (82,
97894), (83, 97895), (84, 97896), (85, 97897), (86, 97898), (87, 97899), (88,
97900), (89, 97901), (90, 97902), (91, 97903), (92, 97904), (93, 97905), (94,
97906), (95, 97907), (96, 97908), (97, 97909), (98, 97910), (99, 97911), (100,
97912), (101, 97913), (102, 97914), (103, 97915), (104, 97916), (105, 97917),
(106, 97918), (107, 97919), (108, 97920), (109, 97921), (110, 97922), (111,
97923), (112, 97924), (113, 97925), (114, 97926), (115, 97927), (116, 97928),
(117, 97929), (118, 97930), (119, 97931), (120, 97932), (121, 97933), (122,
97934), (123, 97935), (124, 97936), (125, 97937), (126, 97938), (127, 97939),
(128, 97940), (129, 97941), (130, 97942), (131, 97943), (132, 97944), (133,
97945), (134, 97946), (135, 97947), (136, 97948), (137, 97949), (138, 97950),
(139, 97951), (140, 97952), (141, 97953), (142, 97954), (143, 97955), (144,
97956), (145, 97957), (146, 97958), (147, 97959), (148, 97960), (149, 97961),
(150, 97962), (151, 97963), (152, 97964), (153, 97965), (154, 97966), (155,
97967), (156, 97968), (157, 97969), (158, 97970), (159, 97971), (160, 97972),
(161, 97973), (162, 97974), (163, 97975), (164, 97976), (165, 97977), (166,
97978), (167, 97979), (168, 97980), (169, 97981), (170, 97982), (171, 97983),
(172, 97984), (173, 97985), (174, 97986), (175, 97987), (176, 97988), (177,
97989), (178, 97990), (179, 97991), (180, 97992), (181, 97993), (182, 97994),
(183, 97995), (184, 97996), (185, 97997), (186, 97998), (187, 97999), (188,
98000), (189, 98001), (190, 98002), (191, 98003), (192, 98004), (193, 98005),
(194, 98006), (195, 98007), (196, 98008), (197, 98009), (198, 98010), (199,
98011), (200, 98012), (201, 98013), (202, 98014), (203, 98015), (204, 98016),
(205, 98017), (206, 98018), (207, 98019), (208, 98020), (209, 98021), (210,
98022), (211, 98023), (212, 98024), (213, 98025), (214, 98026), (215, 98027),
(216, 98028), (217, 98029), (218, 98030), (219, 98031), (220, 98032), (221,
98033), (222, 98034), (223, 98035), (224, 98036), (225, 98037), (226, 98038),
(227, 98039), (228, 98040), (229, 98041), (230, 98042), (231, 98043), (232,
98044), (233, 98045), (234, 98046), (235, 98047), (236, 98048), (237, 98049),
(238, 98050), (239, 98051), (240, 98052), (241, 98053), (242, 98054), (243,
98055), (244, 98056), (245, 98057), (246, 98058), (247, 98059), (248, 98060),
(249, 98061), (250, 98062), (251, 98063), (252, 98064), (253, 98065), (254,
98066), (255, 98067), (256, 98068), (257, 98069), (258, 98070), (259, 98071),
(260, 98072), (261, 98073), (262, 98074), (263, 98075), (264, 98076), (265,
98077), (266, 98078), (267, 98079), (268, 98080), (269, 98081), (270, 98082),
(271, 98083), (272, 98084), (273, 98085), (274, 98086), (275, 98087), (276,
98088), (277, 98089), (278, 98090), (279, 98091), (280, 98092), (281, 98093),
(282, 98094), (283, 98095), (284, 98096), (285, 98097), (286, 98098), (287,
98099), (288, 98100), (289, 98101), (290, 98102), (291, 98103), (292, 98104),
```

```
(293, 98105), (294, 98106), (295, 98107), (296, 98108), (297, 98109), (298,
98110), (299, 98111), (300, 98112), (301, 98113), (302, 98114), (303, 98115),
(304, 98116), (305, 98117), (306, 98118), (307, 98119), (308, 98120), (309,
98121), (310, 98122), (311, 98123), (312, 98124), (313, 98125), (314, 98126),
(315, 98127), (316, 98128), (317, 98129), (318, 98130), (319, 98131), (320,
98132), (321, 98133), (322, 98134), (323, 98135), (324, 98136), (325, 98137),
(326, 98138), (327, 98139), (328, 98140), (329, 98141), (330, 98142), (331,
98143), (332, 98144), (333, 98145), (334, 98146), (335, 98147), (336, 98148),
(337, 98149), (338, 98150), (339, 98151), (340, 98152), (341, 98153), (342,
98154), (343, 98155), (344, 98156), (345, 98157), (346, 98158), (347, 98159),
(348, 98160), (349, 98161), (350, 98162), (351, 98163), (352, 98164), (353,
98165), (354, 98166), (355, 98167), (356, 98168), (357, 98169), (358, 98170),
(359, 98171), (360, 98172), (361, 98173), (362, 98174), (363, 98175), (364,
98176), (365, 98177), (366, 98178), (367, 98179), (368, 98180), (369, 98181),
(370, 98182), (371, 98183), (372, 98184), (373, 98185), (374, 98186), (375,
98187), (376, 98188), (377, 98189), (378, 98190), (379, 98191), (380, 98192),
(381, 98193), (382, 98194), (383, 98195), (384, 98196), (385, 98197), (386,
98198), (387, 98199), (388, 98200), (389, 98201), (390, 98202), (391, 98203),
(392, 98204), (393, 98205), (394, 98206), (395, 98207), (396, 98208), (397,
98209), (398, 98210), (399, 98211), (400, 98212), (401, 98213), (402, 98214),
(403, 98215), (404, 98216), (405, 98217), (406, 98218), (407, 98219), (408,
98220), (409, 98221), (410, 98222), (411, 98223), (412, 98224), (413, 98225),
(414, 98226), (415, 98227), (416, 98228), (417, 98229), (418, 98230), (419,
98231), (420, 98232), (421, 98233), (422, 98234), (423, 98235), (424, 98236),
(425, 98237), (426, 98238), (427, 98239), (428, 98240), (429, 98241), (430,
98242), (431, 98243), (432, 98244), (433, 98245), (434, 98246), (435, 98247),
(436, 98248), (437, 98249), (438, 98250), (439, 98251), (440, 98252), (441,
98253), (442, 98254), (443, 98255), (444, 98256), (445, 98257), (446, 98258),
(447, 98259), (448, 98260), (449, 98261), (450, 98262), (451, 98263), (452,
98264), (453, 98265), (454, 98266), (455, 98267), (456, 98268), (457, 98269),
(458, 98270), (459, 98271), (460, 98272), (461, 98273), (462, 98274), (463,
98275), (464, 98276), (465, 98277), (466, 98278), (467, 98279), (468, 98280),
(469, 98281), (470, 98282), (471, 98283), (472, 98284), (473, 98285), (474,
98286), (475, 98287), (476, 98288), (477, 98289), (478, 98290), (479, 98291),
(480, 98292), (481, 98293), (482, 98294), (483, 98295), (484, 98296), (485,
98297), (486, 98298), (487, 98299), (488, 98300), (489, 98301), (490, 98302),
(491, 98303), (492, 98304), (493, 98305), (494, 98306), (495, 98307), (496,
98308), (497, 98309), (498, 98310), (499, 98311), (500, 98312), (501, 98313),
(502, 98314), (503, 98315), (504, 98316), (505, 98317), (506, 98318), (507,
98319), (508, 98320), (509, 98321), (510, 98322), (511, 98323), (512, 98324),
(513, 98325), (514, 98326), (515, 98327), (516, 98328), (517, 98329), (518,
98330), (519, 98331), (520, 98332), (521, 98333), (522, 98334), (523, 98335),
```

```
(524, 98336), (525, 98337), (526, 98338), (527, 98339), (528, 98340), (529,
98341), (530, 98342), (531, 98343), (532, 98344), (533, 98345), (534, 98346),
(535, 98347), (536, 98348), (537, 98349), (538, 98350), (539, 98351), (540,
98352), (541, 98353), (542, 98354), (543, 98355), (544, 98356), (545, 98357),
(546, 98358), (547, 98359), (548, 98360), (549, 98361), (550, 98362), (551,
98363), (552, 98364), (553, 98365), (554, 98366), (555, 98367), (556, 98368),
(557, 98369), (558, 98370), (559, 98371), (560, 98372), (561, 98373), (562,
98374), (563, 98375), (564, 98376), (565, 98377), (566, 98378), (567, 98379),
(568, 98380), (569, 98381), (570, 98382), (571, 98383), (572, 98384), (573,
98385), (574, 98386), (575, 98387), (576, 98388), (577, 98389), (578, 98390),
(579, 98391), (580, 98392), (581, 98393), (582, 98394), (583, 98395), (584,
98396), (585, 98397), (586, 98398), (587, 98399), (588, 98400), (589, 98401),
(590, 98402), (591, 9
```

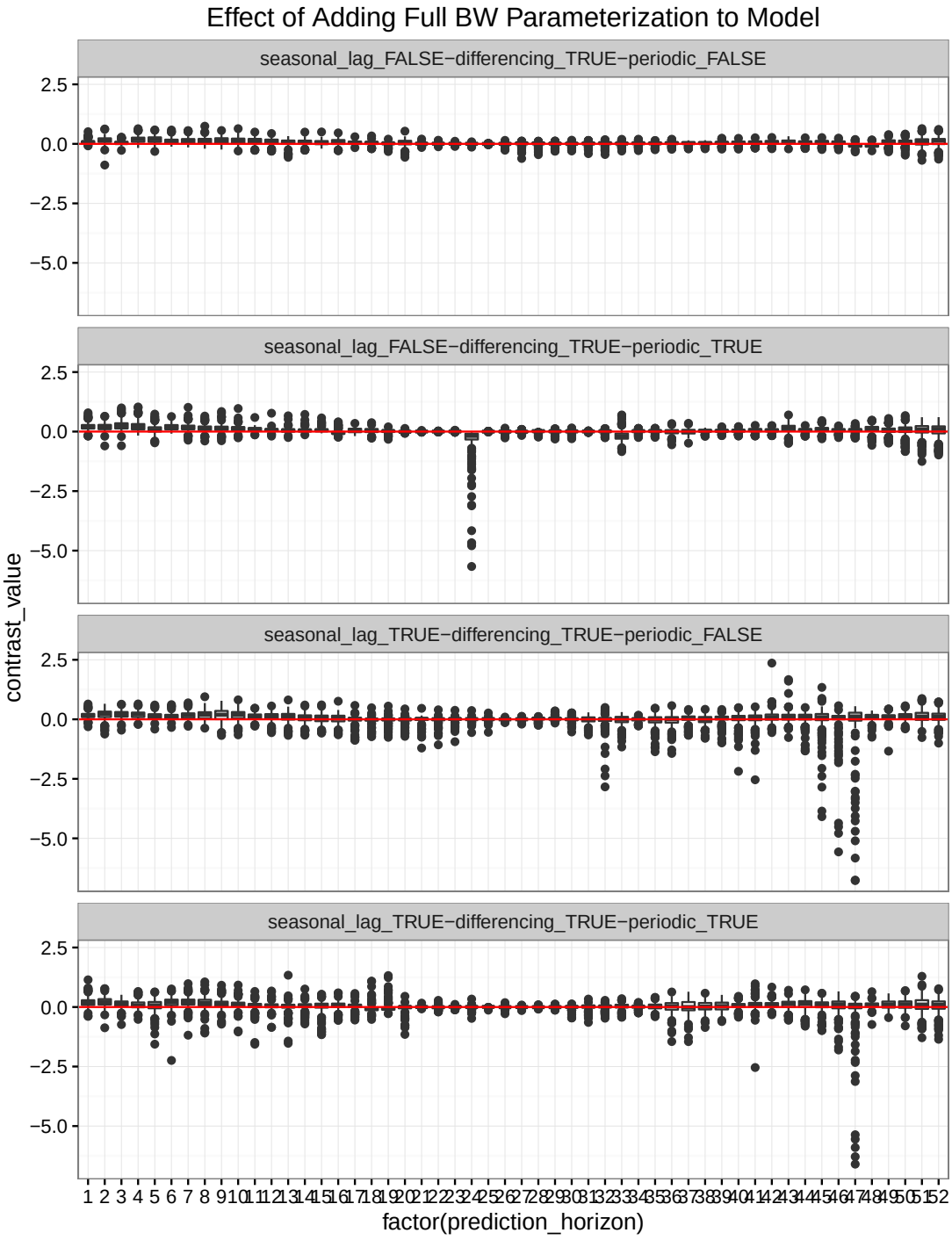

Effect of Adding Periodic Kernel to Model
seasonal_lag_FALSE−differencing_TRUE−bw_full_FALSE
seasonal_lag_FALSE−differencing_TRUE−bw_full_TRUE
seasonal_lag_TRUE−differencing_TRUE−bw_full_FALSE
seasonal_lag_TRUE−differencing_TRUE−bw_full_TRUE
contrast_value
6
4
2
0
−2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
factor(prediction_horizon)

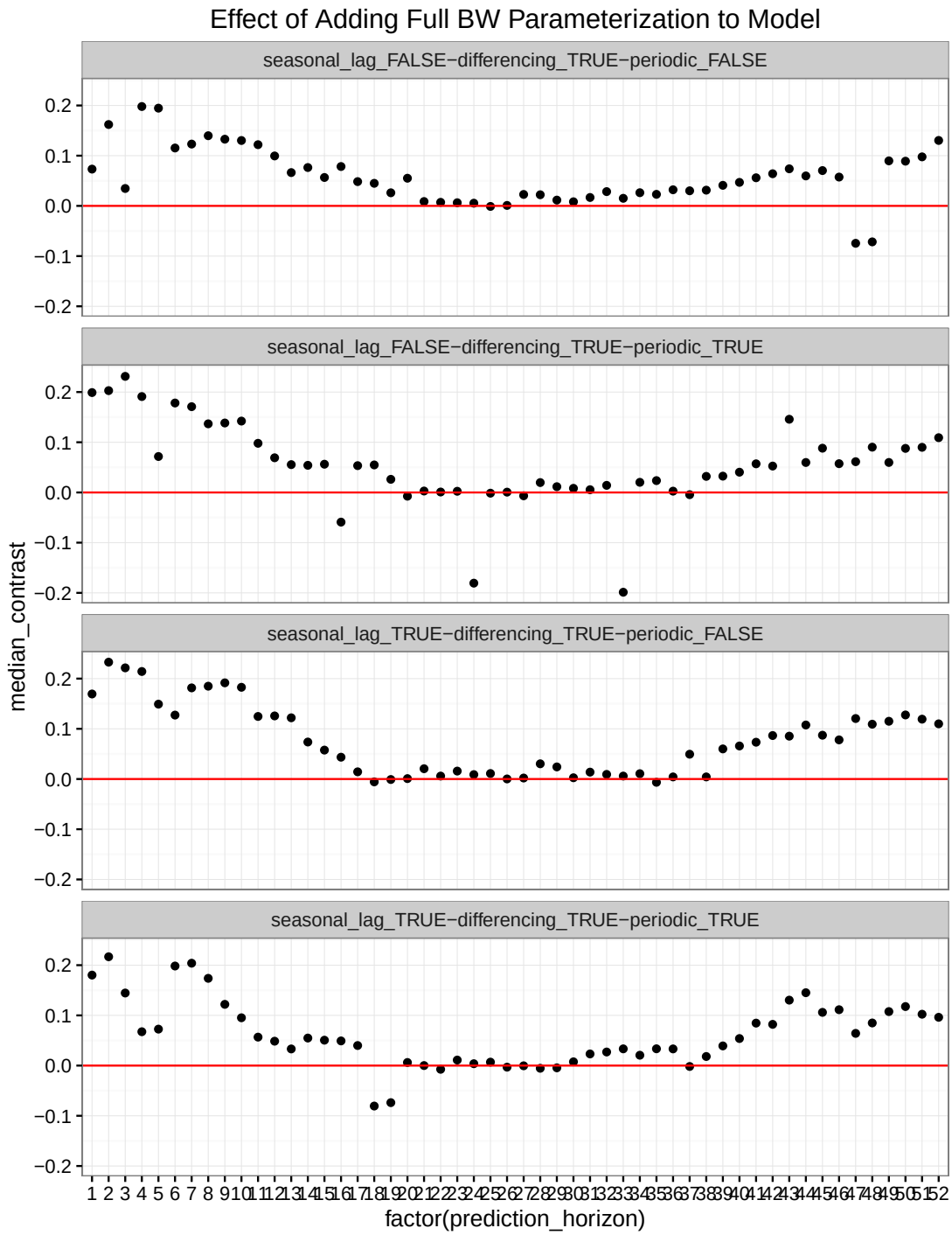

Effect of Adding Periodic Kernel to Model
seasonal_lag_FALSE−differencing_TRUE−bw_full_FALSE
seasonal_lag_FALSE−differencing_TRUE−bw_full_TRUE
seasonal_lag_TRUE−differencing_TRUE−bw_full_FALSE
seasonal_lag_TRUE−differencing_TRUE−bw_full_TRUE
median_contrast
0.2
0.1
0.0
−0.1
−0.2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
factor(prediction_horizon)

```
## Error: Duplicate identifiers for rows (1, 97813), (2, 97814), (3, 97815),
(4, 97816), (5, 97817), (6, 97818), (7, 97819), (8, 97820), (9, 97821), (10,
97822), (11, 97823), (12, 97824), (13, 97825), (14, 97826), (15, 97827), (16,
97828), (17, 97829), (18, 97830), (19, 97831), (20, 97832), (21, 97833), (22,
97834), (23, 97835), (24, 97836), (25, 97837), (26, 97838), (27, 97839), (28,
97840), (29, 97841), (30, 97842), (31, 97843), (32, 97844), (33, 97845), (34,
97846), (35, 97847), (36, 97848), (37, 97849), (38, 97850), (39, 97851), (40,
97852), (41, 97853), (42, 97854), (43, 97855), (44, 97856), (45, 97857), (46,
97858), (47, 97859), (48, 97860), (49, 97861), (50, 97862), (51, 97863), (52,
97864), (53, 97865), (54, 97866), (55, 97867), (56, 97868), (57, 97869), (58,
97870), (59, 97871), (60, 97872), (61, 97873), (62, 97874), (63, 97875), (64,
97876), (65, 97877), (66, 97878), (67, 97879), (68, 97880), (69, 97881), (70,
97882), (71, 97883), (72, 97884), (73, 97885), (74, 97886), (75, 97887), (76,
97888), (77, 97889), (78, 97890), (79, 97891), (80, 97892), (81, 97893), (82,
97894), (83, 97895), (84, 97896), (85, 97897), (86, 97898), (87, 97899), (88,
97900), (89, 97901), (90, 97902), (91, 97903), (92, 97904), (93, 97905), (94,
97906), (95, 97907), (96, 97908), (97, 97909), (98, 97910), (99, 97911), (100,
97912), (101, 97913), (102, 97914), (103, 97915), (104, 97916), (105, 97917),
(106, 97918), (107, 97919), (108, 97920), (109, 97921), (110, 97922), (111,
97923), (112, 97924), (113, 97925), (114, 97926), (115, 97927), (116, 97928),
(117, 97929), (118, 97930), (119, 97931), (120, 97932), (121, 97933), (122,
97934), (123, 97935), (124, 97936), (125, 97937), (126, 97938), (127, 97939),
(128, 97940), (129, 97941), (130, 97942), (131, 97943), (132, 97944), (133,
97945), (134, 97946), (135, 97947), (136, 97948), (137, 97949), (138, 97950),
(139, 97951), (140, 97952), (141, 97953), (142, 97954), (143, 97955), (144,
97956), (145, 97957), (146, 97958), (147, 97959), (148, 97960), (149, 97961),
(150, 97962), (151, 97963), (152, 97964), (153, 97965), (154, 97966), (155,
97967), (156, 97968), (157, 97969), (158, 97970), (159, 97971), (160, 97972),
(161, 97973), (162, 97974), (163, 97975), (164, 97976), (165, 97977), (166,
97978), (167, 97979), (168, 97980), (169, 97981), (170, 97982), (171, 97983),
(172, 97984), (173, 97985), (174, 97986), (175, 97987), (176, 97988), (177,
97989), (178, 97990), (179, 97991), (180, 97992), (181, 97993), (182, 97994),
(183, 97995), (184, 97996), (185, 97997), (186, 97998), (187, 97999), (188,
98000), (189, 98001), (190, 98002), (191, 98003), (192, 98004), (193, 98005),
(194, 98006), (195, 98007), (196, 98008), (197, 98009), (198, 98010), (199,
98011), (200, 98012), (201, 98013), (202, 98014), (203, 98015), (204, 98016),
(205, 98017), (206, 98018), (207, 98019), (208, 98020), (209, 98021), (210,
98022), (211, 98023), (212, 98024), (213, 98025), (214, 98026), (215, 98027),
(216, 98028), (217, 98029), (218, 98030), (219, 98031), (220, 98032), (221,
98033), (222, 98034), (223, 98035), (224, 98036), (225, 98037), (226, 98038),
(227, 98039), (228, 98040), (229, 98041), (230, 98042), (231, 98043), (232,
98044), (233, 98045), (234, 98046), (235, 98047), (236, 98048), (237, 98049),
```

```
(238, 98050), (239, 98051), (240, 98052), (241, 98053), (242, 98054), (243,
98055), (244, 98056), (245, 98057), (246, 98058), (247, 98059), (248, 98060),
(249, 98061), (250, 98062), (251, 98063), (252, 98064), (253, 98065), (254,
98066), (255, 98067), (256, 98068), (257, 98069), (258, 98070), (259, 98071),
(260, 98072), (261, 98073), (262, 98074), (263, 98075), (264, 98076), (265,
98077), (266, 98078), (267, 98079), (268, 98080), (269, 98081), (270, 98082),
(271, 98083), (272, 98084), (273, 98085), (274, 98086), (275, 98087), (276,
98088), (277, 98089), (278, 98090), (279, 98091), (280, 98092), (281, 98093),
(282, 98094), (283, 98095), (284, 98096), (285, 98097), (286, 98098), (287,
98099), (288, 98100), (289, 98101), (290, 98102), (291, 98103), (292, 98104),
(293, 98105), (294, 98106), (295, 98107), (296, 98108), (297, 98109), (298,
98110), (299, 98111), (300, 98112), (301, 98113), (302, 98114), (303, 98115),
(304, 98116), (305, 98117), (306, 98118), (307, 98119), (308, 98120), (309,
98121), (310, 98122), (311, 98123), (312, 98124), (313, 98125), (314, 98126),
(315, 98127), (316, 98128), (317, 98129), (318, 98130), (319, 98131), (320,
98132), (321, 98133), (322, 98134), (323, 98135), (324, 98136), (325, 98137),
(326, 98138), (327, 98139), (328, 98140), (329, 98141), (330, 98142), (331,
98143), (332, 98144), (333, 98145), (334, 98146), (335, 98147), (336, 98148),
(337, 98149), (338, 98150), (339, 98151), (340, 98152), (341, 98153), (342,
98154), (343, 98155), (344, 98156), (345, 98157), (346, 98158), (347, 98159),
(348, 98160), (349, 98161), (350, 98162), (351, 98163), (352, 98164), (353,
98165), (354, 98166), (355, 98167), (356, 98168), (357, 98169), (358, 98170),
(359, 98171), (360, 98172), (361, 98173), (362, 98174), (363, 98175), (364,
98176), (365, 98177), (366, 98178), (367, 98179), (368, 98180), (369, 98181),
(370, 98182), (371, 98183), (372, 98184), (373, 98185), (374, 98186), (375,
98187), (376, 98188), (377, 98189), (378, 98190), (379, 98191), (380, 98192),
(381, 98193), (382, 98194), (383, 98195), (384, 98196), (385, 98197), (386,
98198), (387, 98199), (388, 98200), (389, 98201), (390, 98202), (391, 98203),
(392, 98204), (393, 98205), (394, 98206), (395, 98207), (396, 98208), (397,
98209), (398, 98210), (399, 98211), (400, 98212), (401, 98213), (402, 98214),
(403, 98215), (404, 98216), (405, 98217), (406, 98218), (407, 98219), (408,
98220), (409, 98221), (410, 98222), (411, 98223), (412, 98224), (413, 98225),
(414, 98226), (415, 98227), (416, 98228), (417, 98229), (418, 98230), (419,
98231), (420, 98232), (421, 98233), (422, 98234), (423, 98235), (424, 98236),
(425, 98237), (426, 98238), (427, 98239), (428, 98240), (429, 98241), (430,
98242), (431, 98243), (432, 98244), (433, 98245), (434, 98246), (435, 98247),
(436, 98248), (437, 98249), (438, 98250), (439, 98251), (440, 98252), (441,
98253), (442, 98254), (443, 98255), (444, 98256), (445, 98257), (446, 98258),
(447, 98259), (448, 98260), (449, 98261), (450, 98262), (451, 98263), (452,
98264), (453, 98265), (454, 98266), (455, 98267), (456, 98268), (457, 98269),
(458, 98270), (459, 98271), (460, 98272), (461, 98273), (462, 98274), (463,
98275), (464, 98276), (465, 98277), (466, 98278), (467, 98279), (468, 98280),
```

```
(469, 98281), (470, 98282), (471, 98283), (472, 98284), (473, 98285), (474,
98286), (475, 98287), (476, 98288), (477, 98289), (478, 98290), (479, 98291),
(480, 98292), (481, 98293), (482, 98294), (483, 98295), (484, 98296), (485,
98297), (486, 98298), (487, 98299), (488, 98300), (489, 98301), (490, 98302),
(491, 98303), (492, 98304), (493, 98305), (494, 98306), (495, 98307), (496,
98308), (497, 98309), (498, 98310), (499, 98311), (500, 98312), (501, 98313),
(502, 98314), (503, 98315), (504, 98316), (505, 98317), (506, 98318), (507,
98319), (508, 98320), (509, 98321), (510, 98322), (511, 98323), (512, 98324),
(513, 98325), (514, 98326), (515, 98327), (516, 98328), (517, 98329), (518,
98330), (519, 98331), (520, 98332), (521, 98333), (522, 98334), (523, 98335),
(524, 98336), (525, 98337), (526, 98338), (527, 98339), (528, 98340), (529,
98341), (530, 98342), (531, 98343), (532, 98344), (533, 98345), (534, 98346),
(535, 98347), (536, 98348), (537, 98349), (538, 98350), (539, 98351), (540,
98352), (541, 98353), (542, 98354), (543, 98355), (544, 98356), (545, 98357),
(546, 98358), (547, 98359), (548, 98360), (549, 98361), (550, 98362), (551,
98363), (552, 98364), (553, 98365), (554, 98366), (555, 98367), (556, 98368),
(557, 98369), (558, 98370), (559, 98371), (560, 98372), (561, 98373), (562,
98374), (563, 98375), (564, 98376), (565, 98377), (566, 98378), (567, 98379),
(568, 98380), (569, 98381), (570, 98382), (571, 98383), (572, 98384), (573,
98385), (574, 98386), (575, 98387), (576, 98388), (577, 98389), (578, 98390),
(579, 98391), (580, 98392), (581, 98393), (582, 98394), (583, 98395), (584,
98396), (585, 98397), (586, 98398), (587, 98399), (588, 98400), (589, 98401),
(590, 98402), (591, 9
```


Effect of Adding Full BW Parameterization to Model
seasonal_lag_FALSE−differencing_TRUE−periodic_FALSE
seasonal_lag_FALSE−differencing_TRUE−periodic_TRUE
seasonal_lag_TRUE−differencing_TRUE−periodic_FALSE
seasonal_lag_TRUE−differencing_TRUE−periodic_TRUE
contrast_value
2.5
0.0
−2.5
−5.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
factor(prediction_horizon)


Effect of Adding Full BW Parameterization to Model
seasonal_lag_FALSE−differencing_TRUE−periodic_FALSE
seasonal_lag_FALSE−differencing_TRUE−periodic_TRUE
seasonal_lag_TRUE−differencing_TRUE−periodic_FALSE
seasonal_lag_TRUE−differencing_TRUE−periodic_TRUE
median_contrast
0.2
0.1
0.0
−0.1
−0.2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
factor(prediction_horizon)

```
pit_inc_traj_acf_by_season <- rbind.fill(lapply(
        seq_len(11),
        function(season_row_ind) {
            temp <- acf(pit_incidence_trajectories[season_row_ind, ], plot = FALSE)
            return(data.frame(
                    season = rownames(pit_incidence_trajectories)[season_row_ind],
                    acf = temp$acf,
                    lag = temp$lag
                ))
        }
    ))
```

```
## Error in as.ts(x): object 'pit_incidence_trajectories' not found
```

```
acf_null_limits <- qnorm((1 + 0.95)/2)/sqrt(max_prediction_horizon)
```

```
## Error in eval(expr, envir, enclos): object 'max_prediction_horizon' not
found
```

```
ggplot(pit_inc_traj_acf_by_season) +
    geom_point(aes(x = lag, y = acf), colour = "grey") +
    geom_line(aes(x = lag, y = acf, group = season), colour = "grey") +
    geom_hline(yintercept = 0) +
    geom_hline(yintercept = acf_null_limits, colour = "blue", linetype = 2) +
    geom_hline(yintercept = -acf_null_limits, colour = "blue", linetype = 2) +
    theme_bw()
```

```
## Error in ggplot(pit_inc_traj_acf_by_season): object 'pit_inc_traj_acf_by_season'
not found
```

```
pit_inc_traj_acf_by_season <- rbind.fill(lapply(
        seq_len(11),
        function(season_row_ind) {
            temp <- acf(pit_incidence_trajectories[season_row_ind, ], plot = FALSE)
            return(data.frame(
                    season = rownames(pit_incidence_trajectories)[season_row_ind],
                    acf = temp$acf,
                    lag = temp$lag,
                    type = "observed",
                    stringsAsFactors = FALSE
                ))
        }
    ))
```

```
## Error in as.ts(x): object 'pit_incidence_trajectories' not found

acf_null_limits <- qnorm((1 + 0.95)/2)/sqrt(max_prediction_horizon)

## Error in eval(expr, envir, enclos): object 'max_prediction_horizon' not
found

ggplot(pit_inc_traj_acf_by_season) +
    geom_point(aes(x = lag, y = acf), colour = "grey") +
    geom_line(aes(x = lag, y = acf, group = season), colour = "grey") +
    geom_hline(yintercept = 0) +
    geom_hline(yintercept = acf_null_limits, colour = "blue", linetype = 2) +
    geom_hline(yintercept = -acf_null_limits, colour = "blue", linetype = 2) +
    theme_bw()

## Error in ggplot(pit_inc_traj_acf_by_season): object 'pit_inc_traj_acf_by_season'
not found

n_sim <- 11L

clayton_copula_fit <- fitCopula(
    copula = claytonCopula(dim = max_prediction_horizon),
    data = pit_incidence_trajectories,
    method = "ml")

## Error in eval(expr, envir, enclos): could not find function "fitCopula"

frank_copula_fit <- fitCopula(
    copula = frankCopula(dim = max_prediction_horizon),
    data = pit_incidence_trajectories,
    method = "ml")

## Error in eval(expr, envir, enclos): could not find function "fitCopula"

t_copula_fit <- fitCopula(
    copula = tCopula(dim = max_prediction_horizon, df.fixed = TRUE),
    data = pit_incidence_trajectories,
    method = "ml")

## Error in eval(expr, envir, enclos): could not find function "fitCopula"

ar1_normal_copula_fit <- fitCopula(
    copula = normalCopula(dim = max_prediction_horizon, dispstr = "ar1"),
    data = pit_incidence_trajectories,
    method = "ml")

## Error in eval(expr, envir, enclos): could not find function "fitCopula"
```

```
toep_normal_copula_fit <- fitCopula(
    copula = normalCopula(dim = max_prediction_horizon, dispstr = "toep"),
    data = pit_incidence_trajectories,
    method = "ml")

## Error in eval(expr, envir, enclos): could not find function "fitCopula"

sim_pit_seq_acf <- rbind.fill(lapply(
    c(frank_copula_fit, clayton_copula_fit, t_copula_fit, toep_normal_copula_fit),
    function(copula_fit) {
        rbind.fill(lapply(
            seq_len(n_sim),
            function(sim_ind) {
                predictive_copula <- copula_fit@copula
                ## Note that the variance estimate for parameters is low; at least we're c
                orig_params <- predictive_copula@parameters
                predictive_copula@parameters <- rmvnorm(1, copula_fit@estimate, sigma = co
                random_params <- predictive_copula@parameters
                if(identical(class(predictive_copula)[1], "normalCopula")) {
                    ## randomly generated parameters above may not yield a positive defini
                    while(min(eigen(getSigma(predictive_copula))$values) < 0.00001) {
                        predictive_copula@parameters <- copula:::makePosDef(getSigma(predi
                    }
                }

                sim_sequence <- rCopula(1, predictive_copula)[1, ]

                temp <- acf(sim_sequence, plot = FALSE)
                return(data.frame(
                    sim_ind = sim_ind,
                    acf = temp$acf,
                    lag = temp$lag,
                    type = class(copula_fit@copula)[[1]],
                    stringsAsFactors = FALSE
                ))
            }
        ))
    }
))

## Error in lapply(c(frank_copula_fit, clayton_copula_fit, t_copula_fit, : object
'frank_copula_fit' not found

pit_inc_traj_acf_by_season$sim_ind <- pit_inc_traj_acf_by_season$season
```

```
## Error in eval(expr, envir, enclos): object 'pit_inc_traj_acf_by_season' not
found

combined_acfs <- rbind.fill(pit_inc_traj_acf_by_season, sim_pit_seq_acf)

## Error in rbind.fill(pit_inc_traj_acf_by_season, sim_pit_seq_acf): object
'pit_inc_traj_acf_by_season' not found

ggplot(combined_acfs) +
    geom_point(aes(x = lag, y = acf), colour = "grey") +
    geom_line(aes(x = lag, y = acf, group = sim_ind), colour = "grey") +
    geom_hline(yintercept = 0) +
    geom_hline(yintercept = acf_null_limits, colour = "blue", linetype = 2) +
    geom_hline(yintercept = -acf_null_limits, colour = "blue", linetype = 2) +
    facet_wrap(~type, ncol = 1) +
    theme_bw()

## Error in ggplot(combined_acfs): object 'combined_acfs' not found
```

Future Work

Ensembles – either ensembles of KCDE and/or include as a component in an ensemble Bayesianize?

References

1. John Aitchison and Colin GG Aitken. Multivariate binary discrimination by the kernel method. *Biometrika*, 63(3):413–420, 1976.
2. Tarn Duong and Martin L Hazelton. Cross-validation bandwidth matrices for multivariate kernel density estimation. *Scandinavian Journal of Statistics*, 32(3):485–506, 2005.
3. Jianqing Fan and Tsz Ho Yim. A crossvalidation method for estimating conditional densities. *Biometrika*, 91(4):819–834, 2004.
4. Peter Hall, Jeff Racine, and Qi Li. Cross-validation and the estimation of conditional probability densities. *Journal of the American Statistical Association*, 99(468):1015–1026, 2004.
5. Jeffrey D Hart and Philippe Vieu. Data-driven bandwidth choice for density estimation based on dependent data. *The Annals of Statistics*, 18(2):873–890, 1990.
6. Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Science + Business Media, 2 edition, 2009.
7. Jooyoung Jeon and James W Taylor. Using conditional kernel density estimation for wind power density forecasting. *Journal of the American Statistical Association*, 107(497):66–79, 2012.
8. Qi Li and Jeff Racine. Nonparametric estimation of distributions with categorical and continuous data. *journal of multivariate analysis*, 86(2):266–292, 2003.

9. Qi Li and Jeffrey S Racine. Nonparametric estimation of conditional cdf and quantile functions with mixed categorical and continuous data. *Journal of Business & Economic Statistics*, 26(4):423–434, 2008.
10. Qi Li and Jeffrey Scott Racine. *Nonparametric econometrics: theory and practice*. Princeton University Press, Princeton, NJ, 2006.
11. Desheng Ouyang, Qi Li, and Jeffrey Racine. Cross-validation and the estimation of probability distributions with categorical data. *Journal of Nonparametric Statistics*, 18(1):69–100, 2006.
12. George Sugihara and Robert M. May. Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series, April 1990.
13. Cécile Viboud, Pierre-Yves Boëlle, Fabrice Carrat, Alain-Jacques Valleron, and Antoine Flahault. Prediction of the spread of influenza epidemics by the method of analogues. *American Journal of Epidemiology*, 158(10):996–1006, 2003.

Wang and [v. n Ryzin(1981)]wang1981SmoothEstDiscreteDistn Min-Chiang Wang and John [v]an Ryzin. A class of smooth estimators for discrete distributions. *Biometrika*, 68(1):301–309, 1981.