Journal Title
XX(X):1–36

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
copula, infectious disease, kernel conditional density estimation, prediction

Introduction

Accurate prediction of infectious disease incidence is important for public health officials planning disease prevention and control measures such as vector control and increased use of personal protective equipment by medical personnel during periods of high disease incidence (cite ***). Several quantities have emerged as being of particular utility in making these planning decisions (cite ***); in this article we focus on measures of weekly incidence, the timing of the season peak, and incidence in the peak week. Predictive distributions for these quantities 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 separate predictive distributions for disease incidence in each week of the season, and then combine those marginal distributions using copulas to obtain joint predictive distributions for the trajectory of incidence over the course of multiple weeks. Predictive distributions relating to the timing of and incidence at the peak week can be obtained from this joint predictive distribution for the trajectory of disease incidence. In addition to the novel application of these methods to predicting disease incidence, our contributions include the

[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]*

use of a periodic kernel specification to capture seasonality in disease incidence and a method for obtaining multivariate kernel functions that handle discrete data while allowing for a fully parameterized bandwidth matrix.

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 date (the prediction target) and $\mathbf{X}$ is a vector of predictive variables that we condition on in order to make our prediction. In our example applications, $\mathbf{X}$ includes observations of incidence over several recent time points and variables indicating the time of year at which we are making a prediction; in general, it would be possible to include other variables such as weather covariates.

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[?] , female labor force participation[?] , bond yields and value at risk in financial markets[?] , and wind power[?] 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[?] and influenza[?] . 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[?] . However, KCDE offers the advantage of providing a complete predictive distribution rather than only a point prediction. Methods similar to those we explore in this article can also be formulated in the Bayesian framework. One example along these lines is Zhou et al.[?] , who model the time to arrival of a disease in amphibian populations using Dirichlet processes and copulas.

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. KCDE is distinguished from these alternative approaches in that it makes minimal assumptions about the data generating process. This can be either an advantage and a disadvantage of KCDE. In general, flexible non-parametric methods such as KCDE exhibit low bias but high variance. If they are correctly specified, parametric approaches may achieve reduced variance without introducing bias. On the other hand, because non-parametric approaches such as KCDE make fewer assumptions they may outperform incorrectly specified parametric models. An evaluation of the benefits of an approach such as KCDE is therefore dependent on the particular characteristics of the system being modeled, the data that are available, and the quality of the more structured parametric models that are considered as alternatives. We will return to this point in our conclusions.

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

To our knowledge, all previous authors using kernel methods to estimate multivariate densities involving discrete variables have employed a kernel function that is a product of univariate kernel functions[? ? ? ?] . A variety of functional forms have been proposed for this purpose, ranging from blah to blah to blah[? ? ? ? ?] .

Using a product kernel simplifies the mathemetical formulation of the kernel function when 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, asymptotic analysis and experience with applications have shown that using a multivariate kernel function with a bandwidth parameterization that allows for other orientations can result in improved density estimates in many cases (cite ***). We introduce an approach to allowing for discrete kernels with orientation by discretizing an underlying continuous kernel function.

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 $F_{Y_1,\ldots,Y_D}(y_1, \ldots, y_D) = c(F_{Y_1}(y_1), \ldots, F_{Y_D}(y_D); \boldsymbol{\xi})$. Here $c : [0,1]^D \to [0,1]$ is the copula function depending on parameters $\boldsymbol{\xi}$ and mapping the vector of marginal c.d.f. values to the joint c.d.f. value.

It would be possible to handle this task using just the formulation of KCDE we discussed above, but a direct application of this approach has some limitations. First, the performance of kernel-based density estimation methods scales poorly with the dimension of the random vector whose density is being estimated (cite ***). Second, we have found that different information is available in the data at different prediction horizons. For example, we will demonstrate in our applications below that recently observed incidence is important for making short-term predictions, but terms capturing seasonality are more important for making long-term predictions.

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

In this Section, we give a detailed discussion of our methods. Throughout, 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. We use bold letters to indicate column vectors or matrices; capital letters are random variables and lower case letters are observations of those random variables.

Suppose we observe a measure z_t of disease incidence at each point in time $t = 1, \ldots, T$. At time T, our goal is to obtain a predictive distribution $f(z_{T+1}, \ldots, z_{T+H} | T, z_1, \ldots, z_T)$ for the trajectory of disease incidence over a range of prediction horizons from 1 to H weeks in the future given the time at which we are making the predictions and observed incidence at previous times.

Our model represents this density as follows:

$$f(z_{T+1},\ldots,z_{T+H}|T,z_1,\ldots,z_T) = f(z_{T+1},\ldots,z_{T+H}|T,z_{T-l},\, l\in L) \quad (1)$$

$$c^H\{f^1(z_{T+1}|T+1,z_{T-l},\, l\in L;\boldsymbol{\theta}^1),\ldots,f^H(z_{T+H}|T+H,z_{T-l},\, l\in L;\boldsymbol{\theta}^H);\xi^H\}. \quad (2)$$

Equation (1) formalizes an assumption that all of the information about future incidence contained in the observed history of the disease is captured by the incidence at a few recent time points, with lags in the set L. For now, we leave the set of lags included in L unspecified; in our applications to real disease data below, we will consider several candidate sets of lags. In Equation (2), each $f^h(z_{T+h}|T+h,z_{T-l},\, l\in L;\boldsymbol{\theta}^h)$ is a predictive density for one prediction horizon obtained through KCDE. As the notation suggests, we use a separate parameter vector $\boldsymbol{\theta}^h$ for each prediction horizon, but the same set of lags L at all horizons. This may not be the optimal setup, but exploration of other alternatives is beyond the scope of this article. The function $c^H(\cdot)$ is a copula used to tie these marginal predictive densities together into a joint predictive density.

This model entails several sets of parameters: one set $\boldsymbol{\theta}^h$ for the KCDE fit at each prediction horizon $h=1,\ldots,H$, and another set $\boldsymbol{\xi}^H$ for the copula used to obtain joint distributions for trajectories of length H. Following Joe (cite ***), we pursue a two-stage estimation procedure for these parameters. In the first stage, we estimate the parameters for KCDE separately for each horizon h. Then in the second stage, we estimate the copula parameters while holding the KCDE parameters fixed. In the following subsections, we describe our formulations of KCDE and the copula in more detail and discuss parameter estimation for each of these stages in turn.

KCDE for Predictive Densities at Individual Prediction Horizons

We now discuss the methods we use to obtain the predictive density $f^h(z_{T+h}|T+h,z_{T-l},\, l\in L;\boldsymbol{\theta}^h)$ for disease incidence at a particular horizon h. In order to simplify the notation we define two new variables. $Y_t^h = Z_{t+h}$ represents the prediction target relative to time t. $\mathbf{X}_t^h = (T+h, Z_{t-l_1},\ldots,Z_{t-l_M})$, where $l_1,\ldots,l_M$ are the elements of the set L of lags used for prediction, represents the vector of predictive variables relative to time t. We can use the observed quantities z_t to form the pair $(\mathbf{x}_t^h, y_t^h)$ for all $t = 1+\max_m l_m,\ldots,T-h$; for smaller values of t there are not enough observations before t to form $\mathbf{x}_t^h$ and for larger values of t there are not enough observations after t to form y_t^h. With this notation, the distribution we wish to estimate is $f^h(y_T^h|\mathbf{x}_T^h;\boldsymbol{\theta}^h)$.

In order to do this, we regard $\{(\mathbf{y}_t,\mathbf{x}_t), t = 1+\max_m l_m,\ldots,T-h\}$ as a (dependent) sample from the joint distribution of $(\mathbf{X},Y)$, and estimate the conditional distribution of $Y|\mathbf{X}$ via KCDE. The KCDE estimate is given by

$$\widehat{f}_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \frac{\sum_{t\in\boldsymbol{\tau}} K^{\mathbf{X},\mathbf{Y}}\left\{(\mathbf{x}',\mathbf{y}')',(\mathbf{x}_t',\mathbf{y}_t')';\mathbf{H}^{\mathbf{X},\mathbf{Y}}\right\}}{\sum_{t\in\boldsymbol{\tau}} K^{\mathbf{X}}(\mathbf{x},\mathbf{x}_t;\mathbf{H}^{\mathbf{X}})}. \quad (3)$$

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, $\boldsymbol{\tau}$ typically includes all available time points, but proper subsets are used in the cross-validation procedures we discuss later for parameter estimation.

We will work with a slightly restricted specification of Equation (3) in which the kernel function $K^{\mathbf{X},\mathbf{Y}}$ can be written as the product of $K^{\mathbf{X}}$ and a "conditional kernel" $K^{\mathbf{Y}|\mathbf{X}}$:

$$K^{\mathbf{X},\mathbf{Y}}\left\{(\mathbf{x}',\mathbf{y}')',(\mathbf{x}_t',\mathbf{y}_t')';\mathbf{H}^{\mathbf{X},\mathbf{Y}}\right\} = K^{\mathbf{X}}\left\{\mathbf{x},\mathbf{x}_t;\mathbf{H}^{\mathbf{X}}\right\} K^{\mathbf{Y}|\mathbf{X}}\left\{\mathbf{y},\mathbf{y}_t|\mathbf{x},\mathbf{x}_t;\mathbf{H}^{\mathbf{X},\mathbf{Y}}\right\}. \tag{4}$$

Make clear that H^X is a sub-matrix of $H^{Y,X}$. Hall, Racine, and Li[?] say that this "does not adversely affect the rate of convergence of estimators..."

With this restriction, we can rearrange Equation (3) to obtain

$$\widehat{f}_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) = \sum_{t\in\boldsymbol{\mathcal{T}}} w_t K^{\mathbf{Y}|\mathbf{X}}\left\{\mathbf{y},\mathbf{y}_t|\mathbf{x},\mathbf{x}_t;\mathbf{H}^{\mathbf{X},\mathbf{Y}}\right\}, \text{ where} \tag{5}$$

$$w_t = \frac{K^{\mathbf{X}}\left\{\mathbf{x},\mathbf{x}_t;\mathbf{H}^{\mathbf{X}}\right\}}{\sum_{t^*\in\boldsymbol{\mathcal{T}}} K^{\mathbf{X}}\left\{\mathbf{x},\mathbf{x}_{t^*};\mathbf{H}^{\mathbf{X}}\right\}}. \tag{6}$$

We can now interpret $K^{\mathbf{X}}$ as a weighting function determining how much each observation $(\mathbf{x}_t,\mathbf{y}_t)$ 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. Since the weights w_t sum to 1, this condition ensures that the combined density estimate integrates to 1. The bandwidth matrices $\mathbf{H}^{\mathbf{X}}$ and $\mathbf{H}^{\mathbf{X},\mathbf{Y}}$ are parameterized by $\boldsymbol{\theta}^h$ and control the locality and orientation of the weighting function and the contributions to the density estimate from each observation. In practice, we have used the Cholesky decomposition of

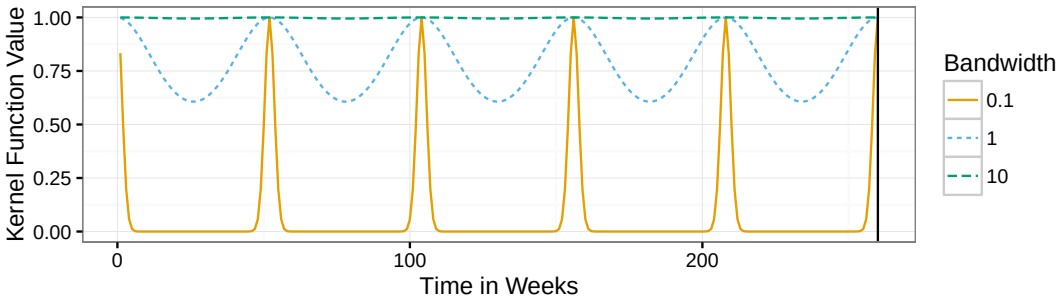
In order to complete the formulation of the estimator given in Equation (3), we must specify the kernel functions.

We obtain the discretized kernel function by integrating an underlying continuous kernel function over the dimensions corresponding to discrete random variables. To make this concrete, consider a J-dimensional random vector $\mathbf{Z} = (\mathbf{Z}^{d'},\mathbf{Z}^{c'})'$ that is partitioned into a J^d-dimensional subvector $\mathbf{Z}^d$ of discrete random variables and a J^c-dimensional subvector $\mathbf{Z}^c$ of continuous random variables. Without loss of generality, we assume that the discrete variables are the first J^d variables. For each component random variable Z_j, $j=1,\ldots,J$, we denote the set of values that Z_j may take by $\mathcal{D}^j \subseteq \mathbb{R}$. In the continuous cases, we take $\mathcal{D}^j = \mathbb{R}$. In the discrete cases, $\mathcal{D}^j$ is a discrete set such as the positive integers. Our definitions could be modified to handle a component random variable whose distribution comprised a combination of discrete and continuous parts; however, this is not required for our applications so we have not pursued that line here.

Let $L(\mathbf{z},\mathbf{z}^*;\mathbf{H})$ be a continuous multivariate kernel function defined on $\prod_{j=1}^{J}\mathcal{E}^j$. For each discrete variable indexed by $j=1,\ldots,J^d$, we associate lower and upper bounds of integration a_{z_j} and b_{z_j} with each value $z_j \in \mathcal{D}^j$. In order to ensure that our final density estimate integrates to 1, we require that these integration bounds form a disjoint cover of $\mathcal{E}^j$ in the sense that $\cup_{z_j\in\mathcal{D}^j}[a_{z_j},b_{z_j}) = \mathcal{E}^j$ and $\cap_{z_j\in\mathcal{D}^j}[a_{z_j},b_{z_j}) = \emptyset$, the empty set. At a vector of values $\mathbf{z}\in\prod_{j=1}^{J}\mathcal{D}^j$, we define the partially discretized kernel as follows:

$$K(\mathbf{z},\mathbf{z}^*;\mathbf{H}) = \int_{a_{z_1}}^{b_{z_1}} \cdots \int_{a_{z_{J^d}}}^{b_{z_{J^d}}} L(\mathbf{z},\mathbf{z}^*;\mathbf{H})\, dz_1 \cdots dz_{J^d}$$

Figure 1. The periodic kernel function illustrated as a function of time in weeks with $\rho = \pi/52$ and three possible values for the bandwidth parameter h.



This approach can be used to discretize any underlying continuous kernel function. In the applications in Section below, we have used a multivariate log-normal kernel function. This is similar to the suggestion of *** of using log-transformed data, but allows us to work with the data on the original scale. Advantages of this kernel specification include automatic handling of the restriction that counts are non-negative, and approximately capturing the long tail in disease incidence that we will illustrate in the applications Section below.

Seasonality is an important characteristic of many infectious disease time series, as we will illustrate in the applications below. Here we describe one approach to capturing this seasonality within the KCDE framework by using a kernel function that is periodic in the observation time t. This periodic kernel function was originally developed in the literature on Gaussian Processes for the purposes of specifying a periodic covariance function **?** , and is defined by

$$K(t, t^*; \rho, h) = \exp\left[-\frac{\sin^2\{\rho(t - t^*)\}}{2h^2}\right]. \tag{7}$$

This functional form can also be obtained by mapping t to the bivariate pair $\{cos(\rho t), sin(\rho t)\}$ and then applying a bivariate Gaussian kernel to that pair. We picture this periodic kernel function in Figure 1.

This kernel function does not integrate to 1, but this does not preclude us from using it for the conditioning variables $\mathbf{X}_t$ that are used to calculate the observation weights w_t in Equation (5). The result is that observations are weighted according to the similarity of the time of year that an observation was collected with the time of year of the analysis time from which we are predicting forward. Effectively, this allows the approach to set the weights w_t to be small when the observation time does not match the observation time from which we are predicting. The strength of this weighting is determined by the bandwidth parameter h.

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,

$$(\widehat{\mathbf{u}}, \widehat{H}^{\mathbf{X}}, \widehat{H}^{\mathbf{Y}}) \approx \underset{(\mathbf{u}, H^{\mathbf{X}}, H^{\mathbf{Y}})}{\operatorname{argmin}} \sum_{t^*=1+P+L}^{T} Q[\mathbf{y}_{t^*}, \widehat{f}(\mathbf{y}|\mathbf{X} = \mathbf{x}_{t^*}; \mathbf{u}, H^{\mathbf{X}}, H^{\mathbf{Y}}, \{(\mathbf{y}_t, \mathbf{x}_t) : t \in \tau_{t^*}\})] \quad (8)$$

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 $\widehat{f}(\mathbf{y}|\mathbf{X} = \mathbf{x}_{t^*}; \mathbf{u}, H^{\mathbf{X}}, H^{\mathbf{Y}}, \{(\mathbf{y}_t, \mathbf{x}_t) : t \in \tau_{t^*}\})$.

Hart and Vieu[?] show that when kernel density estimation is used to estimate a marginal density with dependent observations, leaving out multiple time points around the target time point in cross validation can yield small improvements in the ISE under certain assumptions about the form of the dependence.

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

Combining Marginal Predictive Distributions with Copulas

The approach we take for some of the prediction targets we examine in our applications is to obtain a joint predictive distribution for disease incidence over a sequence of multiple prediction horizons. We do this by using a copula to combine marginal predictive densities for each of those prediction horizons. Specifically, we us the isotropic Gaussian copula implemented in the `R`[?] package `copula`[?] .

This copula function is given by

$$c(u_1, \ldots, u_J; \boldsymbol{\theta}_c) = \Phi_\Sigma(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_J)), \quad (9)$$

where Φ^{-1} is the inverse c.d.f. of a standard univariate Gaussian distribution and Φ_Σ is the c.d.f. of a multivariate Gaussian distribution with mean $\underline{0}$ and covariance matrix Σ. We set $\Sigma = [\sigma^2_{i,j}]$, where

$$\sigma^2_{i,j} = \begin{cases} 1 \text{ if } i = j, \\ \rho_d \text{ if } |i - j| = d \end{cases} \quad (10)$$

Intuitively, ρ_d captures the correlation of incidence at future times that are d weeks apart.

Estimation by maximum likelihood.

Simulation Study

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[?]; 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.

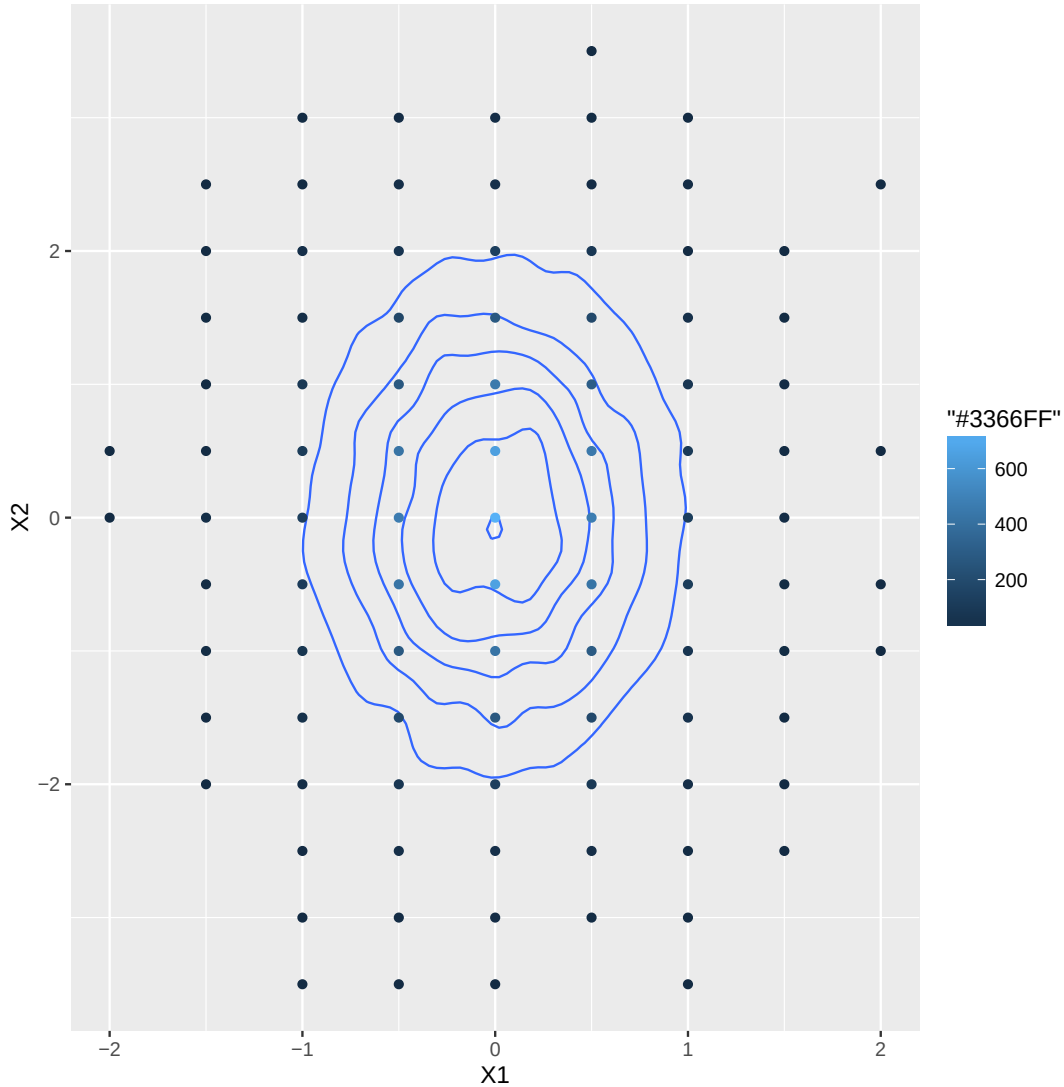
We simulate observations from each of seven distributions. The first five of these are plotted in Figure ***.

```
library(ggplot2)
library(grid)
library(plyr)
library(dplyr)

##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(tidyr)
library(pdtmvn)
library(kcde)

## Warning: failed to assign RegisteredNativeSymbol for logspace_add_C to
logspace_add_C since logspace_add_C is already defined in the 'kcde' namespace
## Warning: failed to assign RegisteredNativeSymbol for
logspace_sum_matrix_rows_C to logspace_sum_matrix_rows_C since
logspace_sum_matrix_rows_C is already defined in the 'kcde' namespace
## Warning: failed to assign RegisteredNativeSymbol for logspace_sub_C to
logspace_sub_C since logspace_sub_C is already defined in the 'kcde' namespace
```

```
## Warning: failed to assign RegisteredNativeSymbol for
logspace_sub_matrix_rows_C to logspace_sub_matrix_rows_C since
logspace_sub_matrix_rows_C is already defined in the 'kcde' namespace
##
## Attaching package: 'kcde'
## The following objects are masked from 'package:pdtmvn':
##
##     logspace_add, logspace_sub, logspace_sub_matrix_rows,
##     logspace_sum_matrix_rows
```

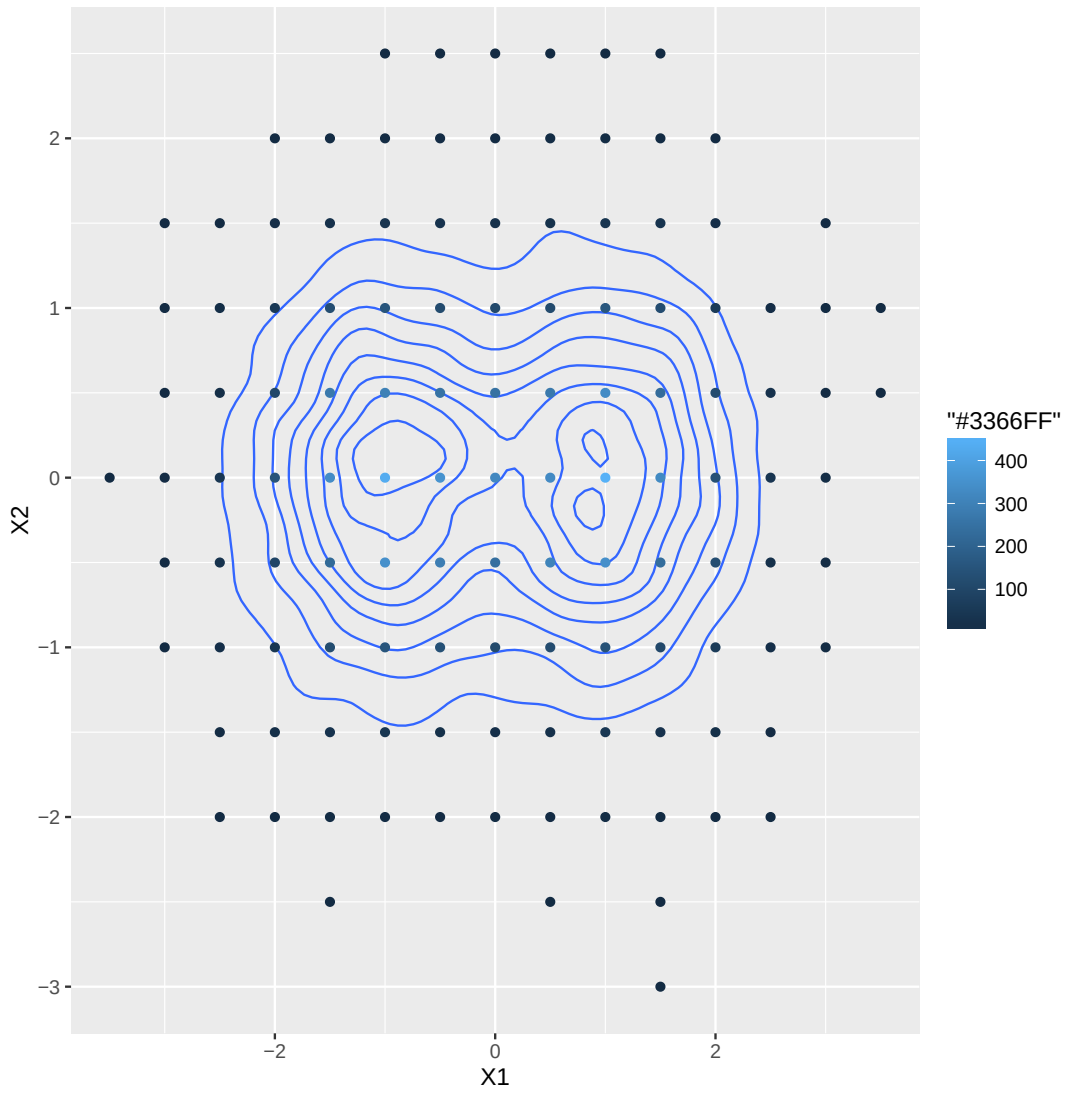
```
source("/media/evan/data/Reich/infectious-disease-prediction-with-kcde/inst/code/sim-densi
```

```
## 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
```

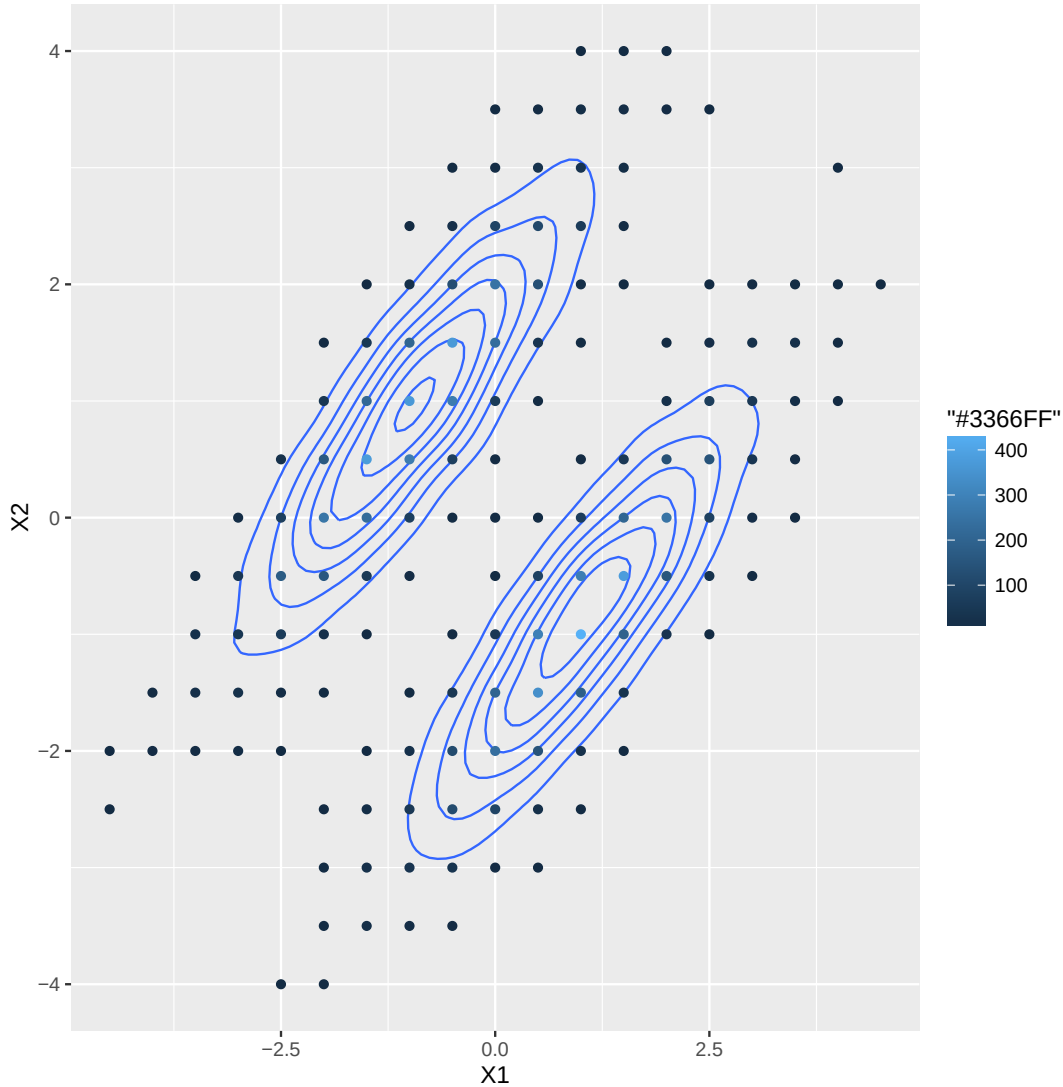



```
## Density family bivariate-B
n_sim <- 10000
discrete_sample <- sim_from_pdtmvn_mixt(n = n_sim, sim_family = "bivariate-B-discretized")
    as.data.frame()
continuous_sample <- sim_from_pdtmvn_mixt(n = n_sim, sim_family = "bivariate-B") %>%
    as.data.frame()
discrete_sample_counts <- discrete_sample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

```
    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
```

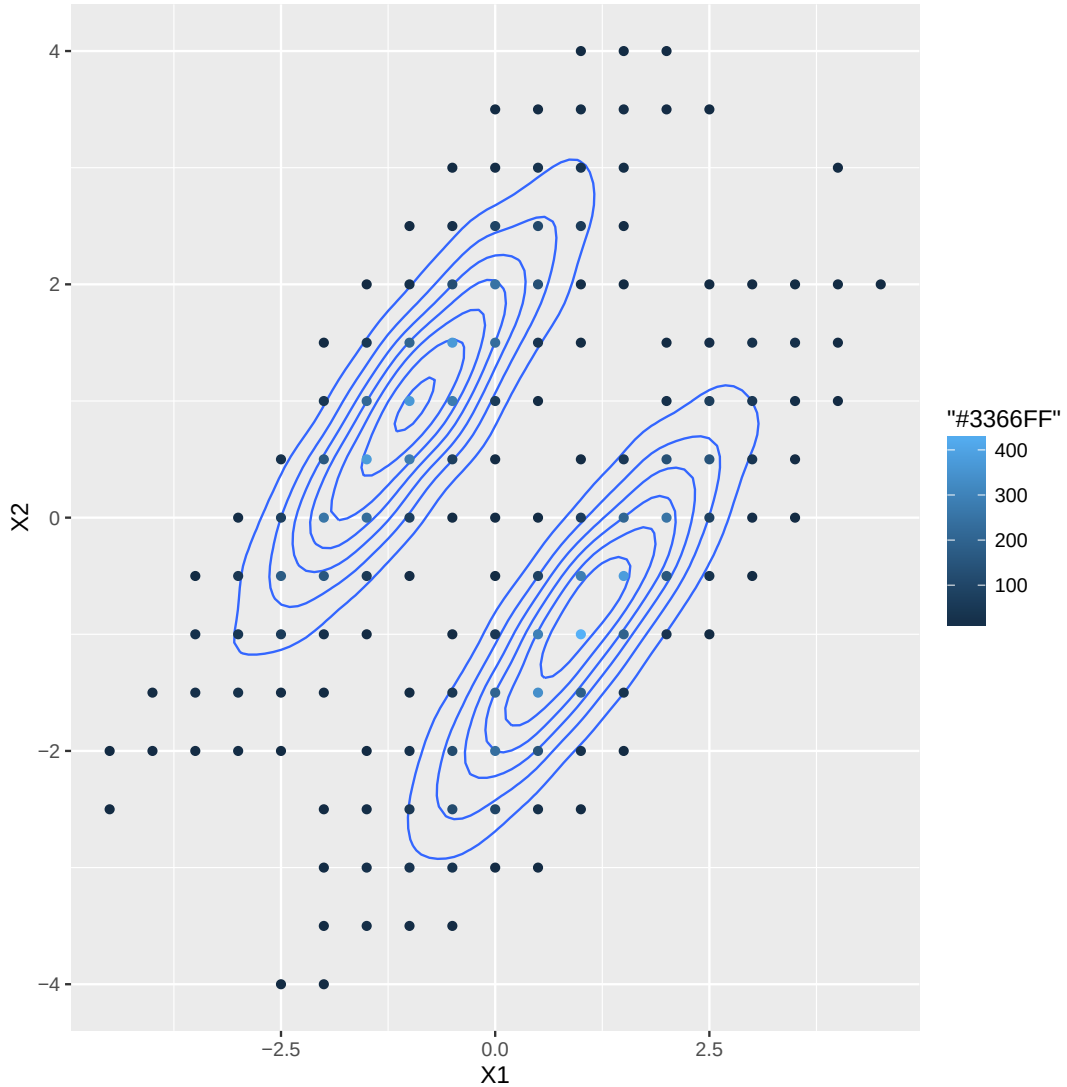


```r
## Density family bivariate-C
n_sim <- 10000
discrete_sample <- sim_from_pdtmvn_mixt(n = n_sim, sim_family = "bivariate-C-discretized") %>%
    as.data.frame()
continuous_sample <- sim_from_pdtmvn_mixt(n = n_sim, sim_family = "bivariate-C") %>%
    as.data.frame()
discrete_sample_counts <- discrete_sample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



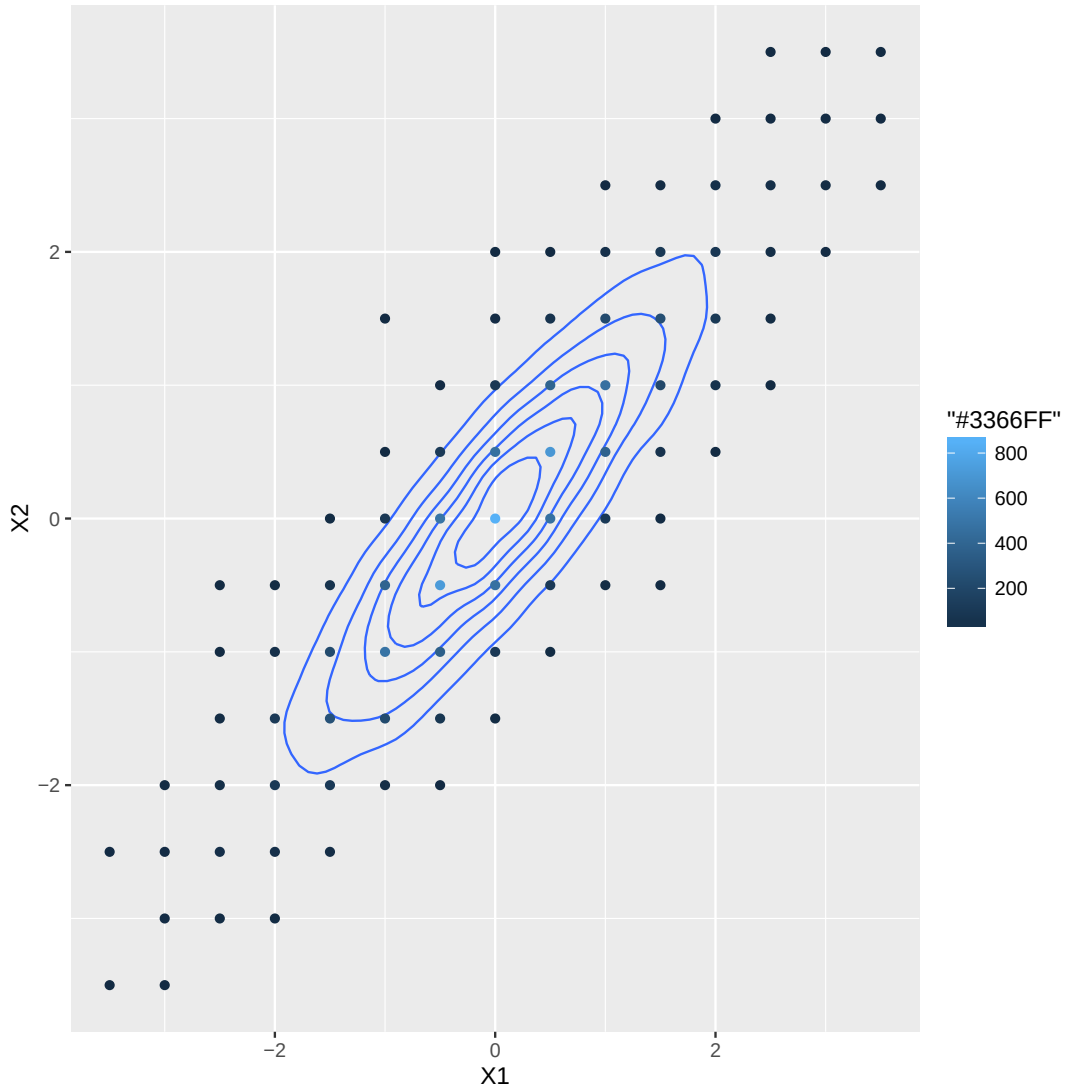

```
## Density family bivariate-D
n_sim <- 10000
discrete_sample <- sim_from_pdtmvn_mixt(n = n_sim, sim_family = "bivariate-D-discretized")
    as.data.frame()

## 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
```



Applications

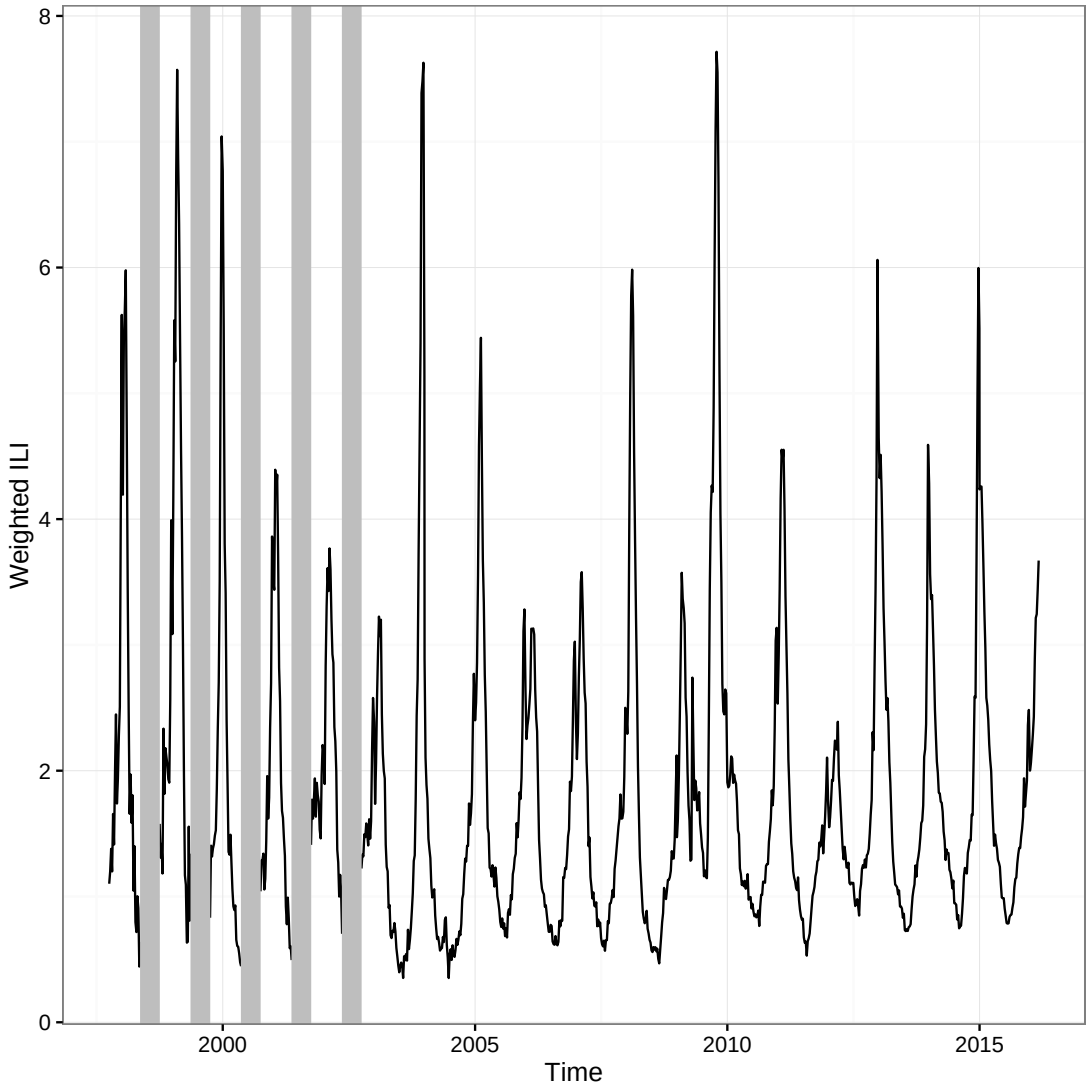
In this Section, we illustrate the methods through applications to prediction in examples with several real time series data sets.

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

```
##
## Attaching package: 'lubridate'
## The following object is masked from 'package:plyr':
##
##     here

##
  |
  |                                                      |   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.

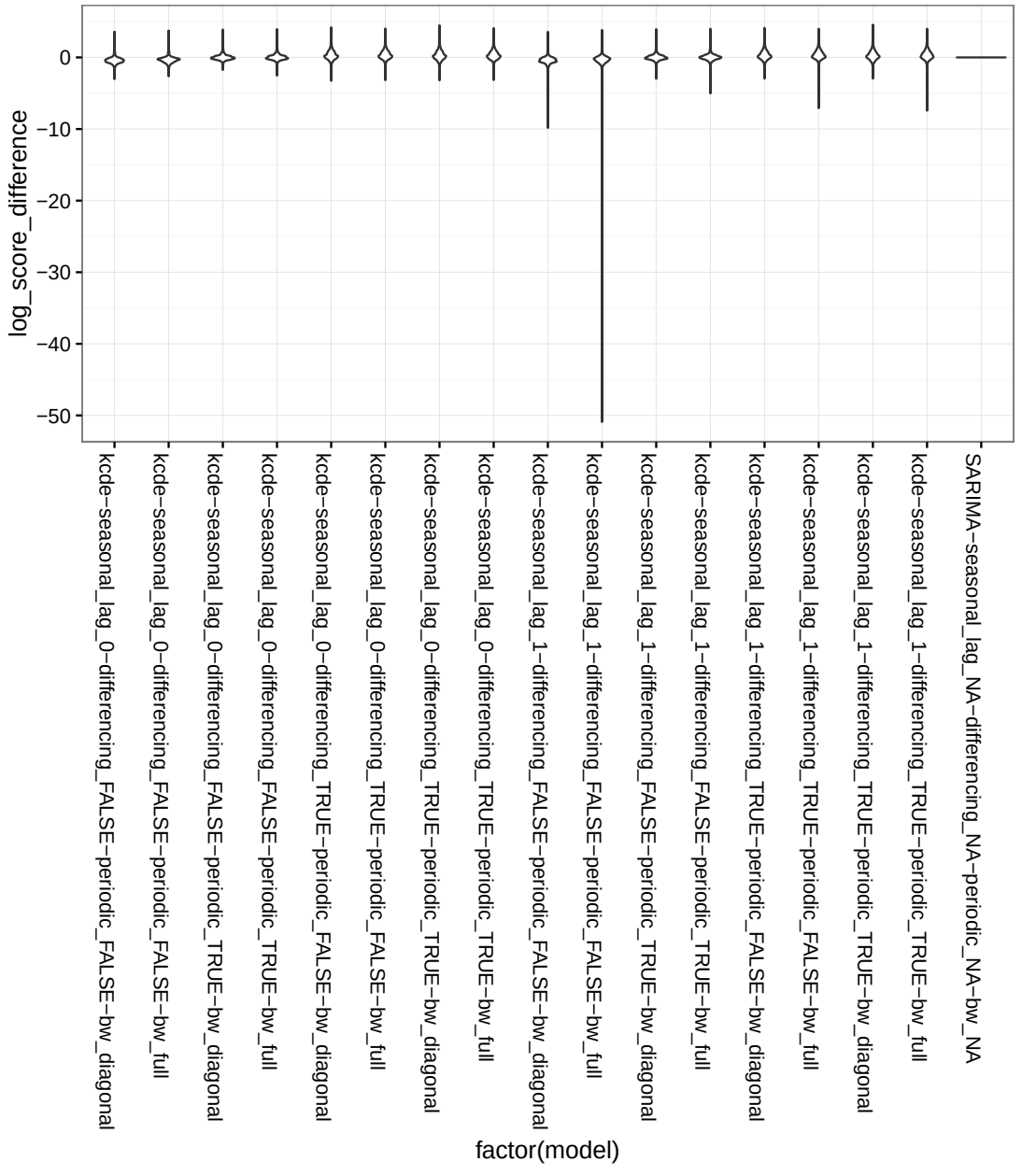


There are several methods that we could employ to handle these missing data:

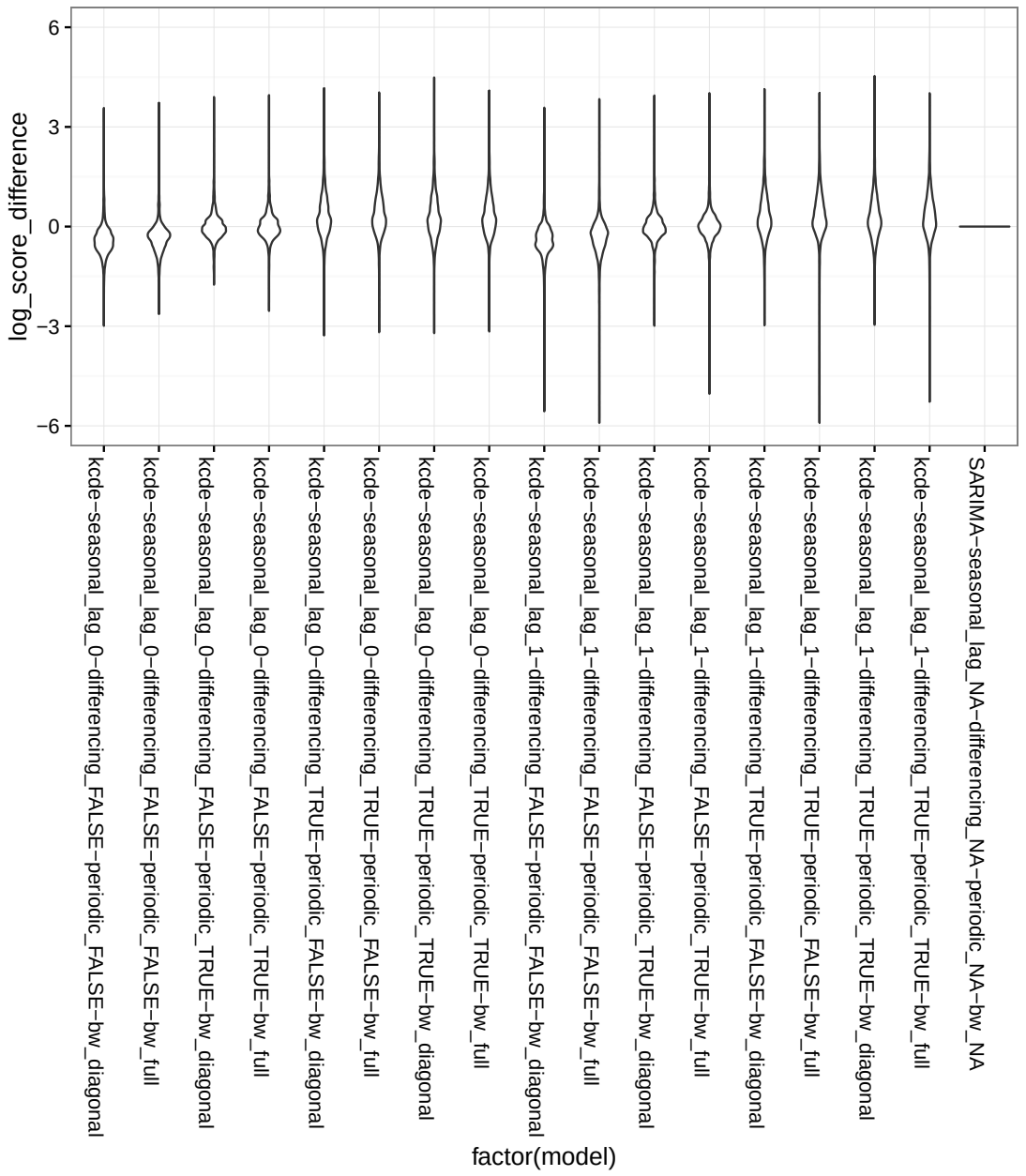
1. Impute the missing values. They are all in the low season, so this should be relatively easy to do.
2. Drop all data up through the last NA.
3. Use the data that are available.

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

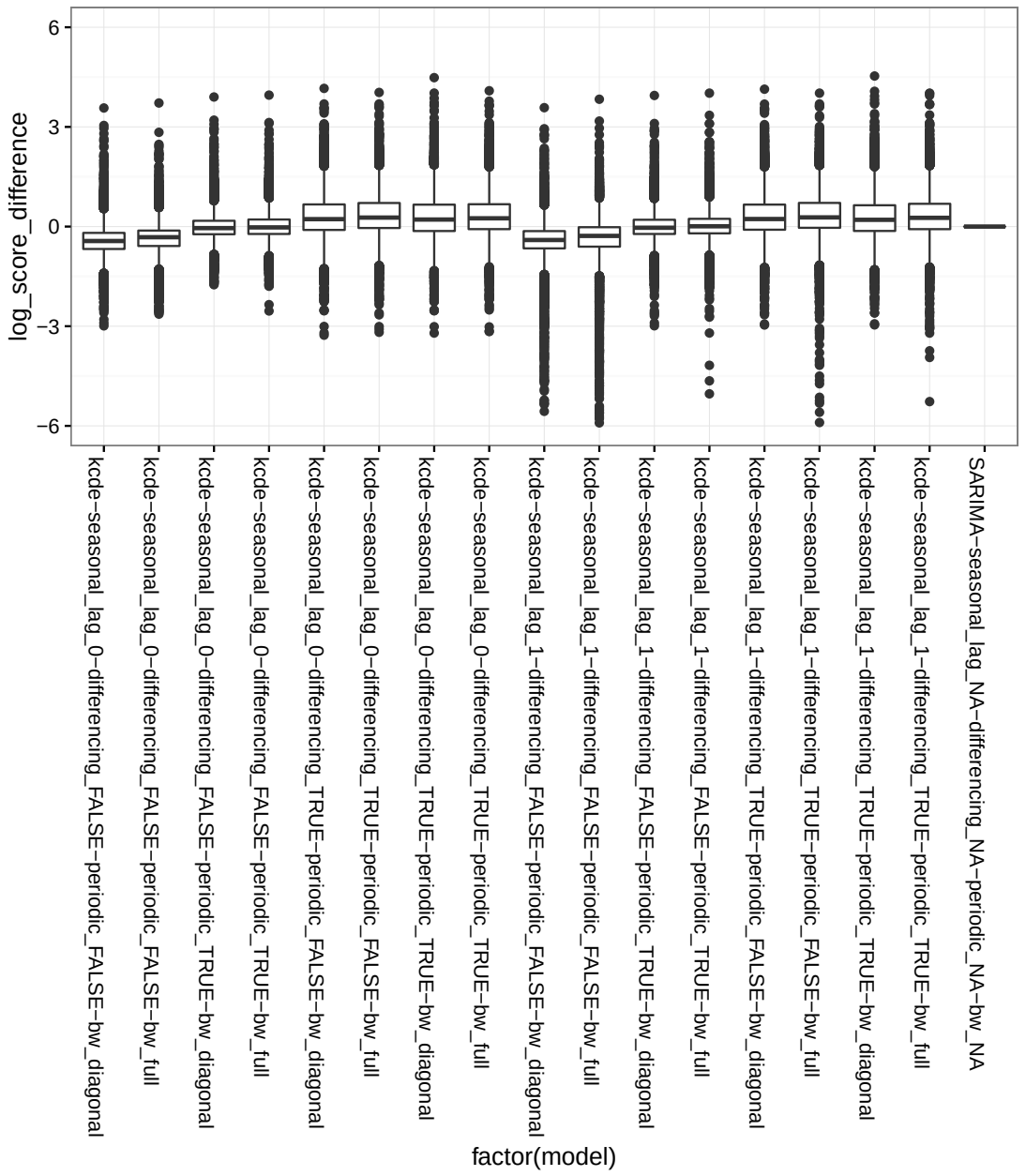
data used in estimation, there will be more examples of prediction of values in the high season using values in the high season and middle of the season than of prediction of values in the high season using values in the low season. This could potentially affect our inference. However, we do not expect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


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

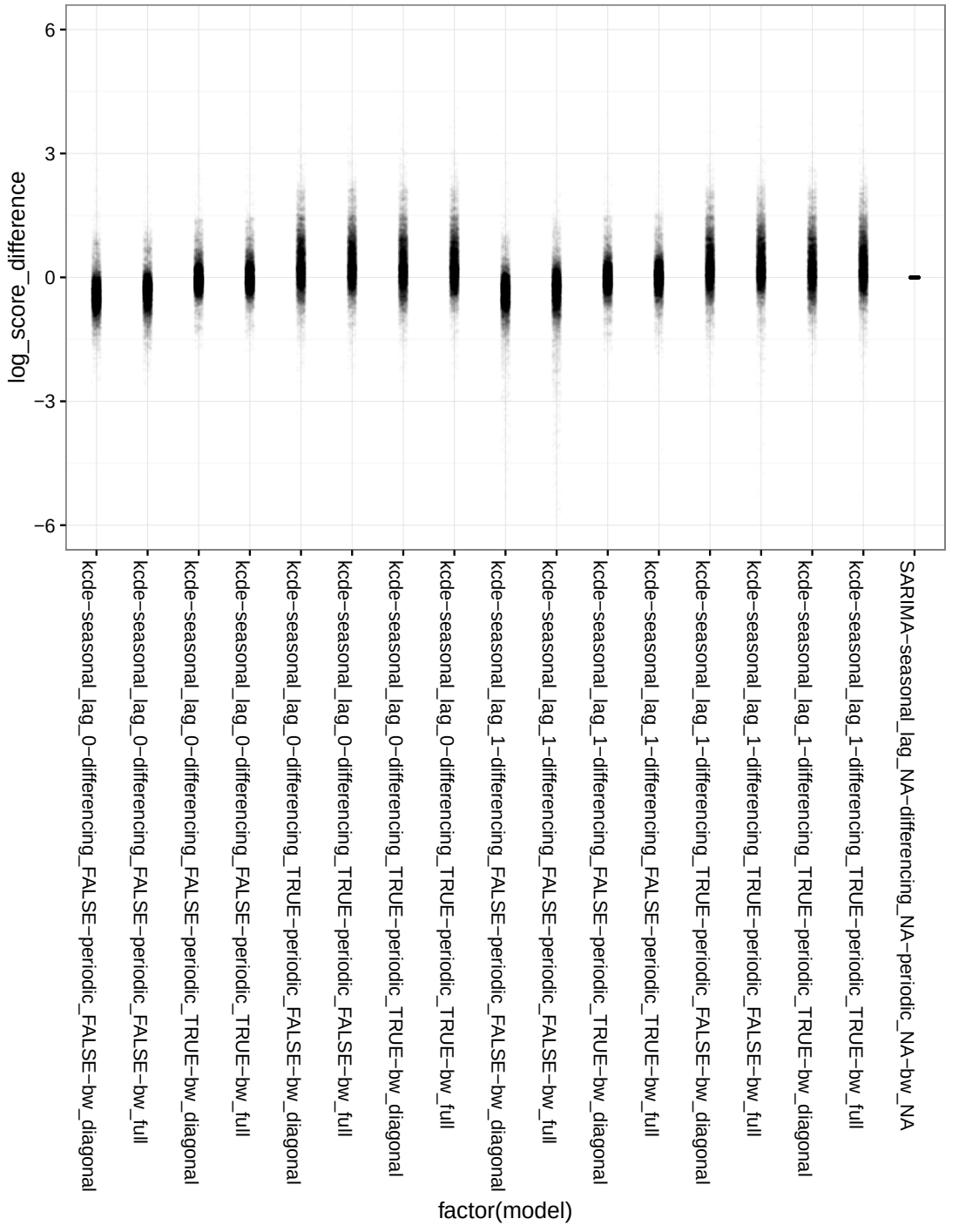
```
## Warning: Removed 7 rows containing non-finite values (stat_ydensity).
```

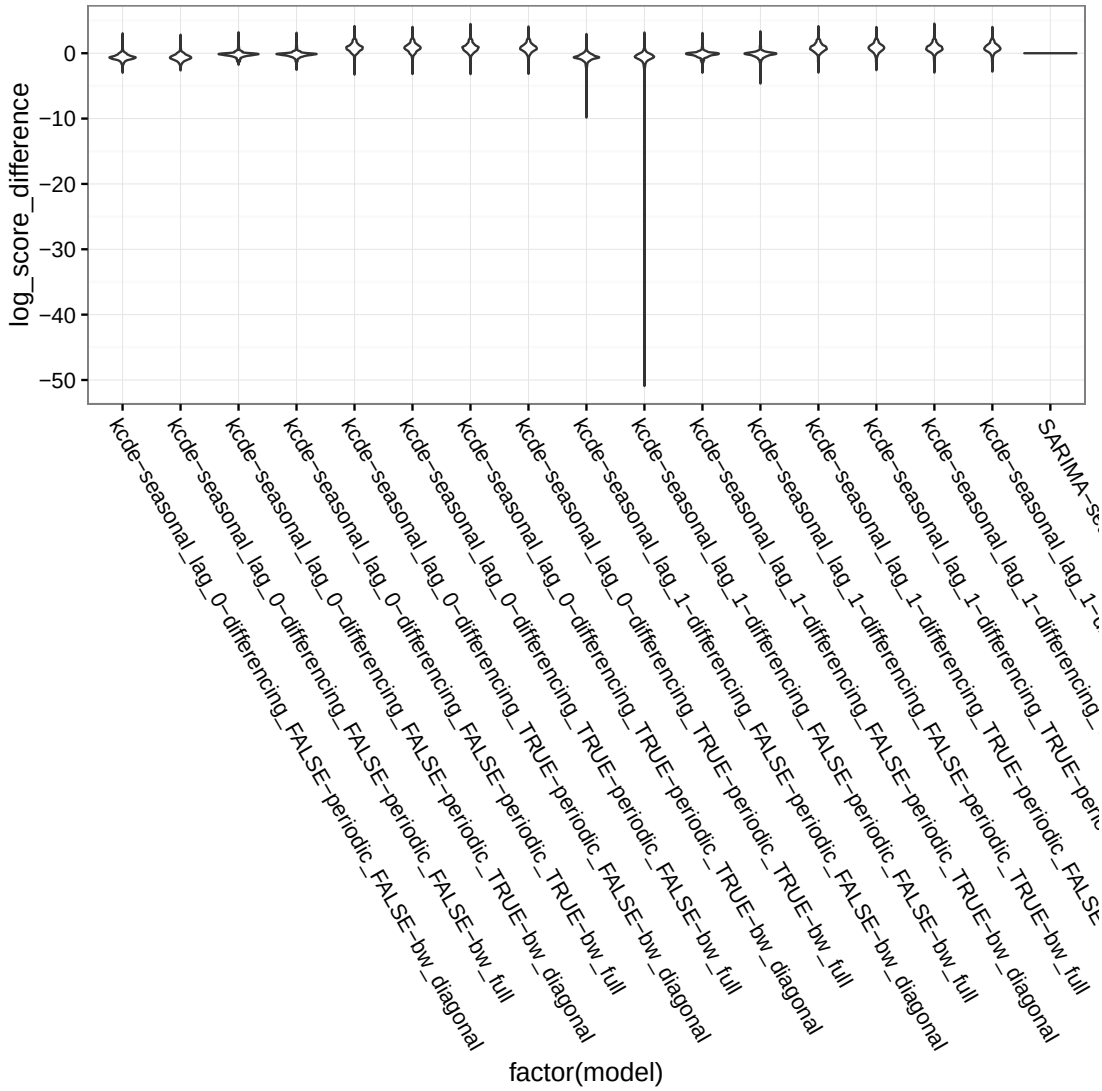


```
## Warning: Removed 7 rows containing non-finite values (stat_boxplot).
```

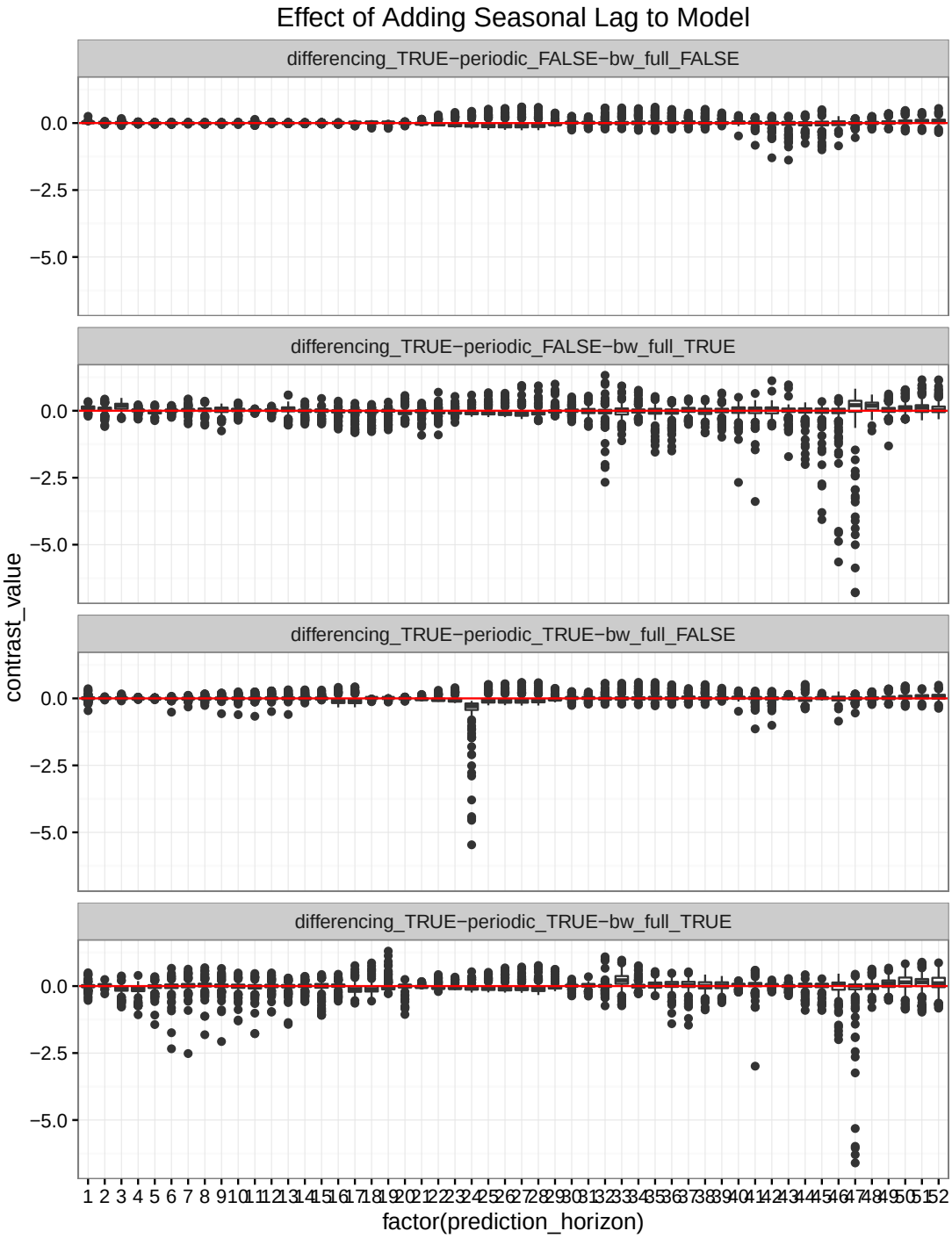


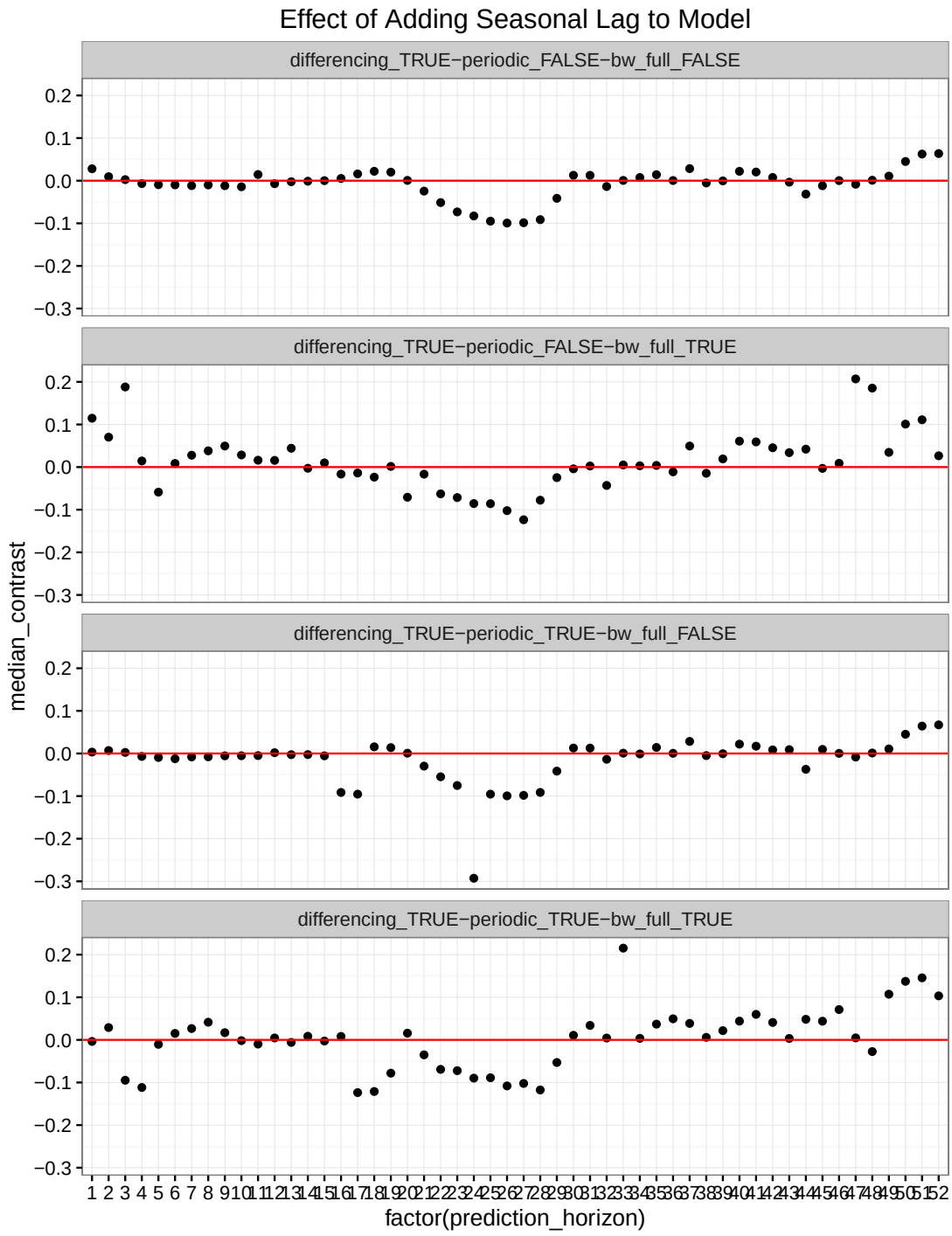
```
## Warning: Removed 7 rows containing missing values (geom_point).
```


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



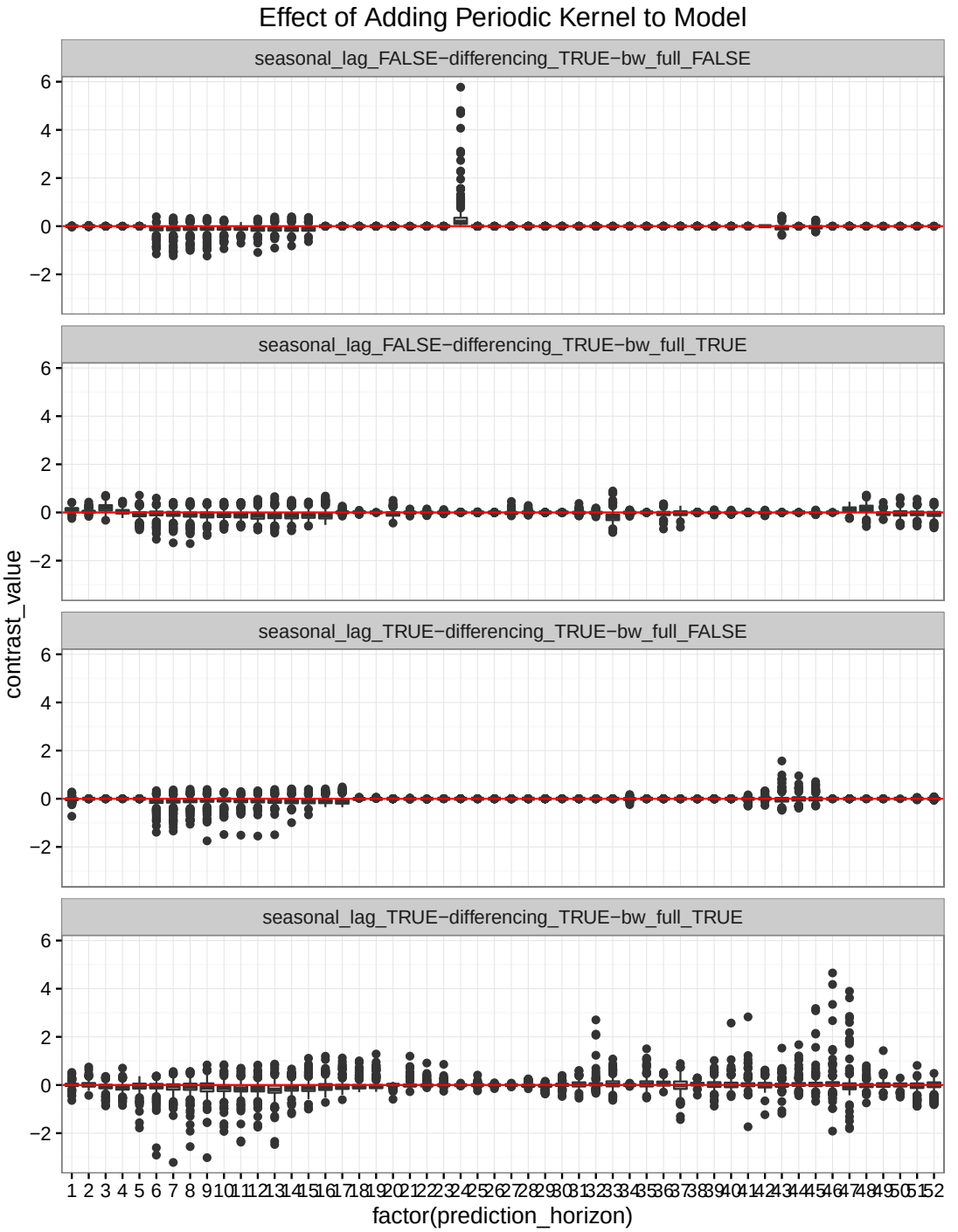

```
## Error in fortify(data): object 'ili_prediction_log_score_diffs' not found
```

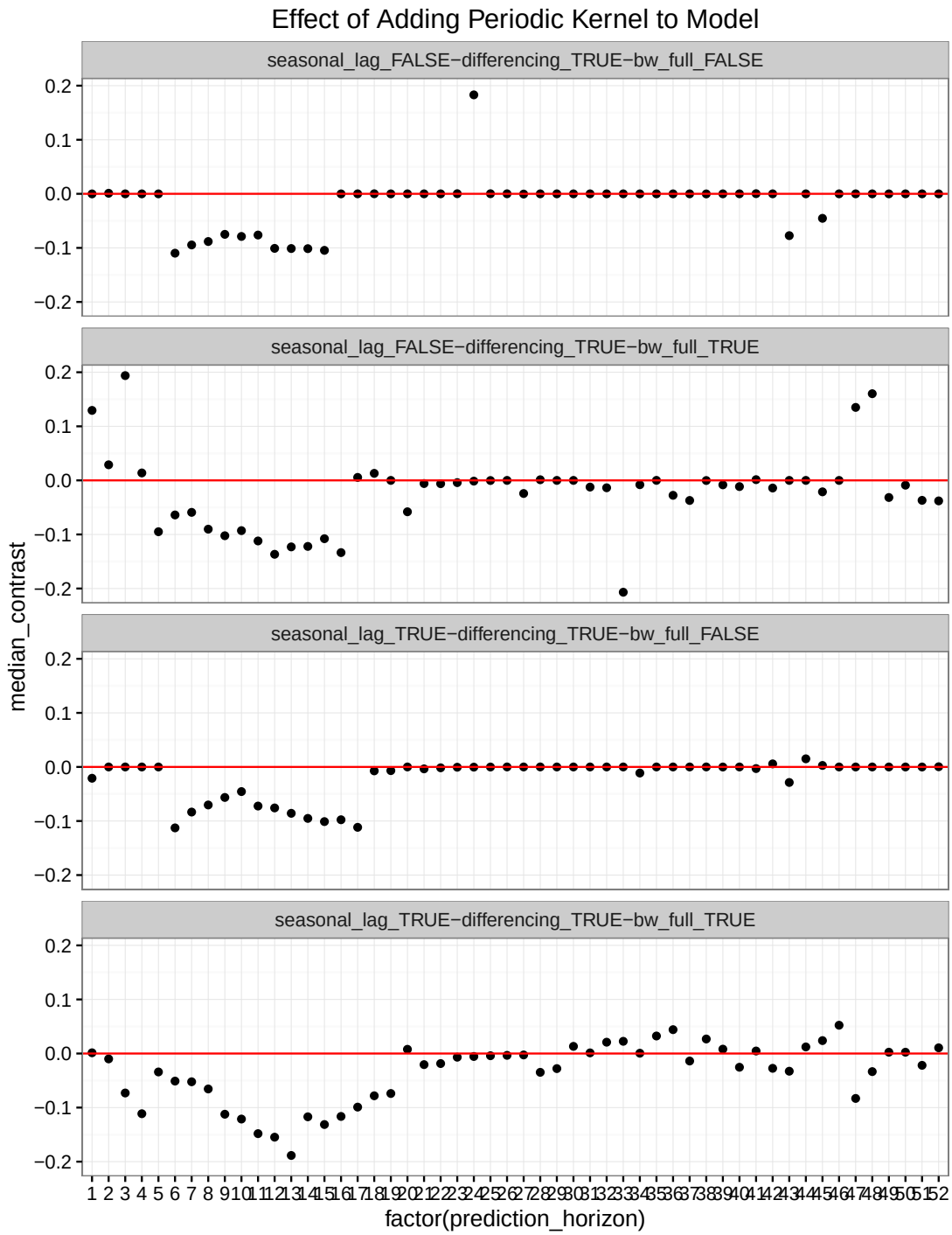

Effect of Adding Seasonal Lag to Model
differencing_TRUE−periodic_FALSE−bw_full_FALSE
differencing_TRUE−periodic_FALSE−bw_full_TRUE
differencing_TRUE−periodic_TRUE−bw_full_FALSE
differencing_TRUE−periodic_TRUE−bw_full_TRUE
contrast_value
0.0
−2.5
−5.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 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)

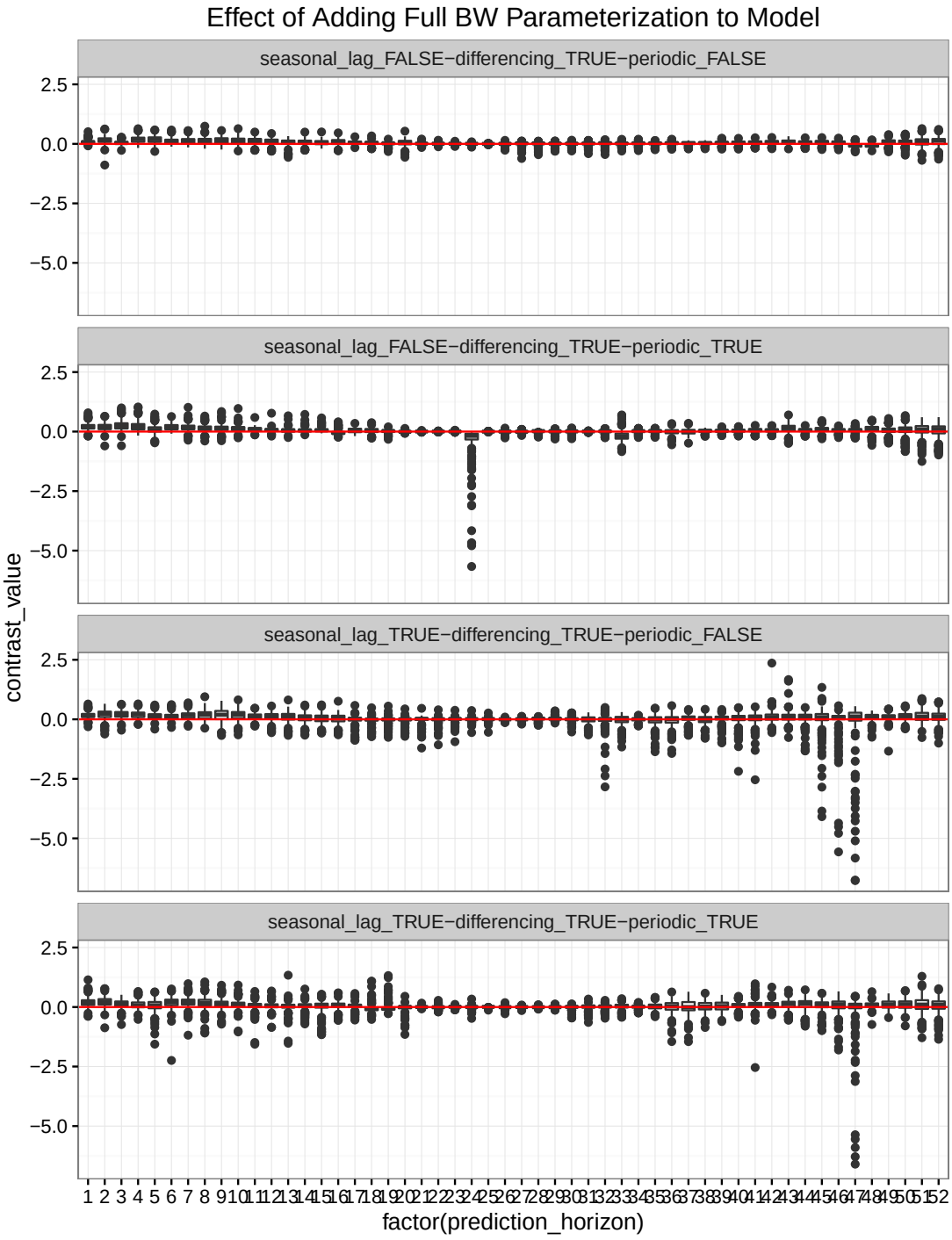

Effect of Adding Seasonal Lag to Model
differencing_TRUE−periodic_FALSE−bw_full_FALSE
differencing_TRUE−periodic_FALSE−bw_full_TRUE
differencing_TRUE−periodic_TRUE−bw_full_FALSE
differencing_TRUE−periodic_TRUE−bw_full_TRUE
median_contrast
0.2
0.1
0.0
−0.1
−0.2
−0.3
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 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)

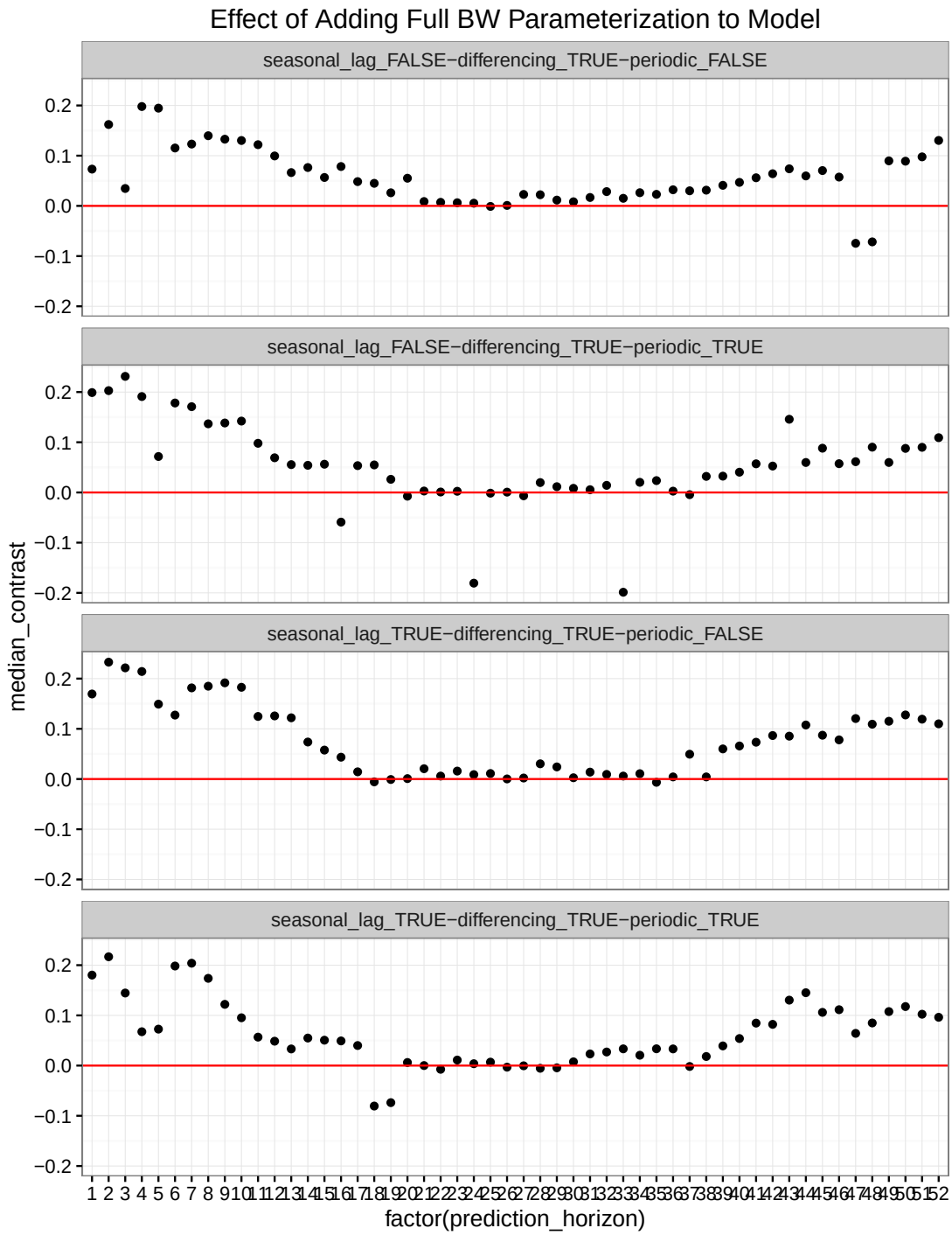
```
## Error in eval(expr, envir, enclos): object 'FALSE' not found
## Error in ggplot(ili_contrast_differencing): object 'ili_contrast_differencing'
not found
```

```
## 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
```


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
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)


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

Hall, Racine, and Li[?] show that when cross-validation is used to select the bandwidth parameters in KCDE using product kernels, the estimated bandwidths corresponding to irrelevant conditioning variables tend to infinity asymptotically as the sample size increases. They discuss the fact that similar results could be obtained for linear combinations of continuous variables if a full bandwidth matrix were used. Our approach for obtaining kernels that can be used with mixed discrete and continuous variables opens up an opportunity to extend this analysis to that case; we have not pursued this mathematical analysis here.

The above results regarding the inclusion of irrelevant conditioning variables hold asymptotically as the sample size increases. However, in practice, data set sizes are often limited. In other modeling settings where some conditioning variables may not be informative, shrinkage methods are often helpful. These methods could be incorporated into a kernel-based approach by imposing a penalty on the elements of the bandwidth matrix; in particular, we suggest that a penalty on the inverse of the bandwidth matrix encouraging it to have small eigenvalues could be helpful. Another alternative would be to pursue the Bayesian framework, using Dirichlet process mixtures with an informative prior on the mixture component covariance matrices.

We could also make some tweaks to our implementation of KCDE. Locally linear – help address edge effects.

Ensembles – either ensembles of KCDE and/or include as a component in an ensemble. Also Bayesian model averaging. Return to discussion of bias/variance trade-off?