1 Introduction

Conditional, mixed continuous and discrete, non-diagonal bw, smoothing, periodic kernel specification

2 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.

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 3, 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:

$$\widehat{f}_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{X}=\mathbf{x}) = \frac{\widehat{f}_{\mathbf{Y},\mathbf{X}}(\mathbf{y},\mathbf{x})}{\widehat{f}_{\mathbf{X}}(\mathbf{x})} \tag{1}$$

$$= \frac{\sum_{t\in\tau} K^{\mathbf{Y},\mathbf{X}}\{(\mathbf{y},\mathbf{x}),(\mathbf{y}_t,\mathbf{x}_t),H^{\mathbf{Y},\mathbf{X}}\}}{\sum_{t\in\tau} K^{\mathbf{X}}(\mathbf{x},\mathbf{x}_t,H^{\mathbf{X}})} \tag{2}$$

$$= \frac{\sum_{t\in\tau} K^{\mathbf{Y}|\mathbf{X}}(\mathbf{y},\mathbf{y}_t|\mathbf{x},\mathbf{x}_t,H^{\mathbf{Y},\mathbf{X}})K^{\mathbf{X}}(\mathbf{x},\mathbf{x}_t,H^{\mathbf{X}})}{\sum_{t\in\tau} K^{\mathbf{X}}(\mathbf{x},\mathbf{x}_t,H^{\mathbf{X}})} \tag{3}$$

$$= \sum_{t\in\tau} w_t K^{\mathbf{Y}|\mathbf{X}}(\mathbf{y},\mathbf{y}_t|\mathbf{x},\mathbf{x}_t,H^{\mathbf{Y},\mathbf{X}}), \text{ where} \tag{4}$$

$$w_t = \frac{K^{\mathbf{X}}(\mathbf{x},\mathbf{x}_t,H^{\mathbf{X}})}{\sum_{t^*\in\tau} K^{\mathbf{X}}(\mathbf{x},\mathbf{x}_{t^*},H^{\mathbf{X}})} \tag{5}$$

In Equation (1), we are simply 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 (3), we obtain separate kernel density estimates for the joint and marginal densities in this quotient. In Equation (4), we rewrite this quotient by passing the denominator of Equation (3) 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 (5), 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 (1) through (5), 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 3.

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:

$$(\widehat{\mathbf{Y}}|\mathbf{X}=\mathbf{x}) = \mathbb{E}_{\widehat{f}_{\mathbf{Y}|\mathbf{X}}}\{\mathbf{Y}|\mathbf{X}=\mathbf{x}\} \tag{6}$$

$$= \int \sum_{t\in\tau} w_t K^{\mathbf{Y}}(\mathbf{y},\mathbf{y}_t,H^{\mathbf{Y}})\mathbf{y}\, d\mathbf{y} \tag{7}$$

$$= \sum_{t\in\tau} w_t \mathbf{y}_t \tag{8}$$

The equality in Equation (8) 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.

3 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,

$$(\widehat{\mathbf{u}}, \widehat{H}^{\mathbf{X}}, \widehat{H}^{\mathbf{Y}}) \approx \underset{(\mathbf{u}, H^{\mathbf{X}}, H^{\mathbf{Y}})}{\operatorname{argmin}} \sum_{t^*=1+P+L}^{T} Q[\mathbf{y}_{t^*}, \widehat{f}(\mathbf{y}|\mathbf{X} = \mathbf{x}_{t^*}; \mathbf{u}, H^{\mathbf{X}}, H^{\mathbf{Y}}, \{(\mathbf{y}_t, \mathbf{x}_t) : t \in \tau_{t^*}\})] \tag{9}$$

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 [Byrd et al.(1995)Byrd, Lu, Nocedal, and Zhu] to estimate the bandwidth parameters. The approximation in Equation (9) is due to the fact that this optimization procedure may not find a global minimum.

4 Examples

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

4.1 Example 1: Influenza Prediction

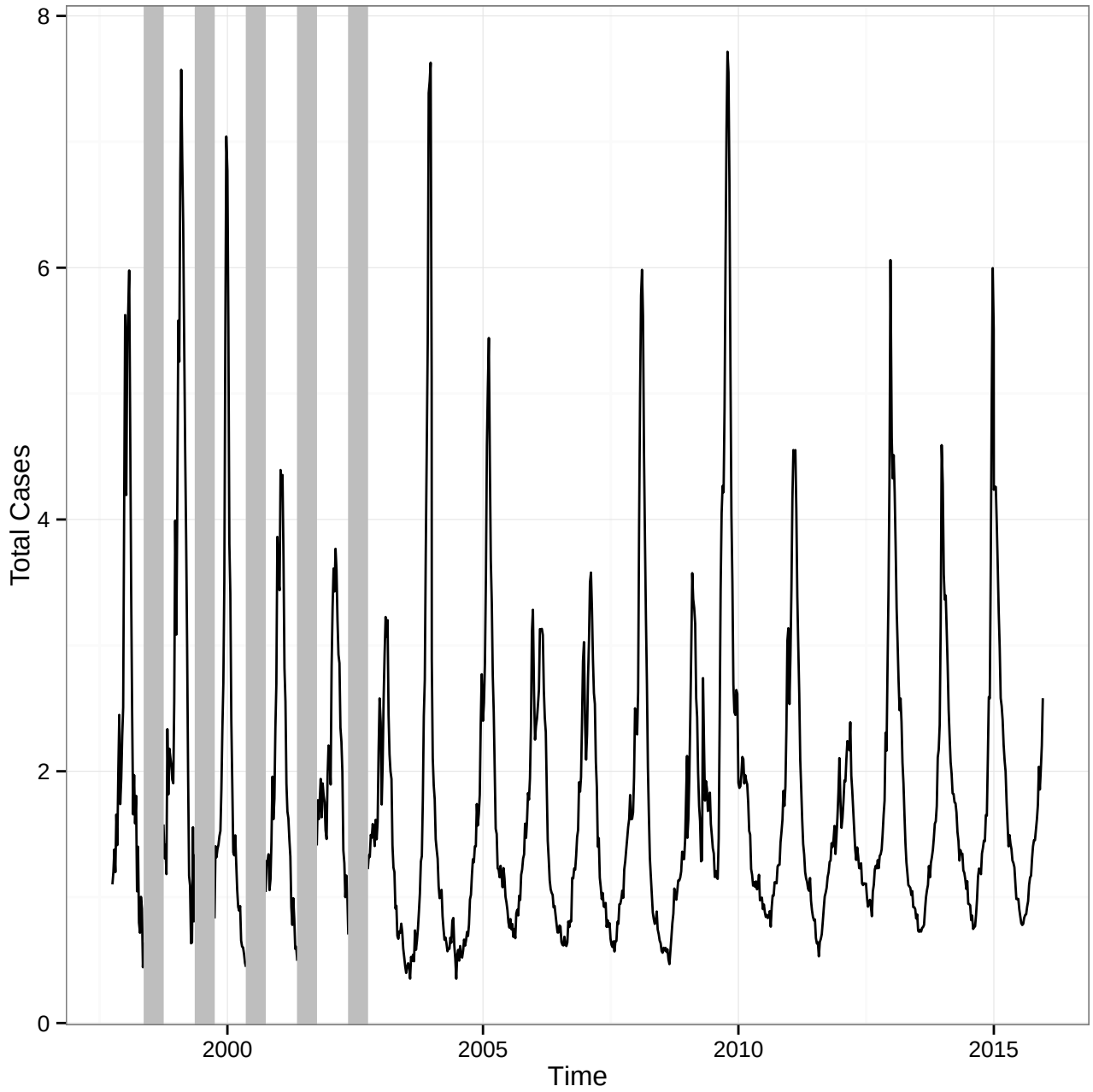
In our first and simplest example, we apply the method for prediction of influenza with prediction horizons of 1 through 4 weeks. Data on influenza incidence are available through `R`'s `cdcfluview` package. Here we create a data set with a nationally aggregated measure of flu incidence

```
library(cdcfluview)
library(dplyr)
library(lubridate)
library(ggplot2)
library(grid)
library(kcde)

usflu<-get_flu_data("national", "ilinet", years=1997:2015)

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

ili_national <- transmute(usflu,
    region.type = REGION.TYPE,
    region = REGION,
    year = YEAR,
    week = WEEK,
    total_cases = as.numeric(X..WEIGHTED.ILI))

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

ili_national$time <- ymd(paste(ili_national$year, "01", "01", sep = "-"))
week(ili_national$time) <- ili_national$week
```

```
ili_national$time_index <- seq_len(nrow(ili_national))

str(ili_national)

## 'data.frame': 952 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 ...
##  $ total_cases: 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.

```
ggplot() +
    geom_line(aes(x = as.Date(time), y = total_cases), data =
ili_national) +
    geom_vline(aes(xintercept = as.numeric(as.Date(time))),
        colour = "grey",
        data = ili_national[is.na(ili_national$total_cases), ]) +
    scale_x_date() +
    xlab("Time") +
    ylab("Total Cases") +
    theme_bw()
```



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

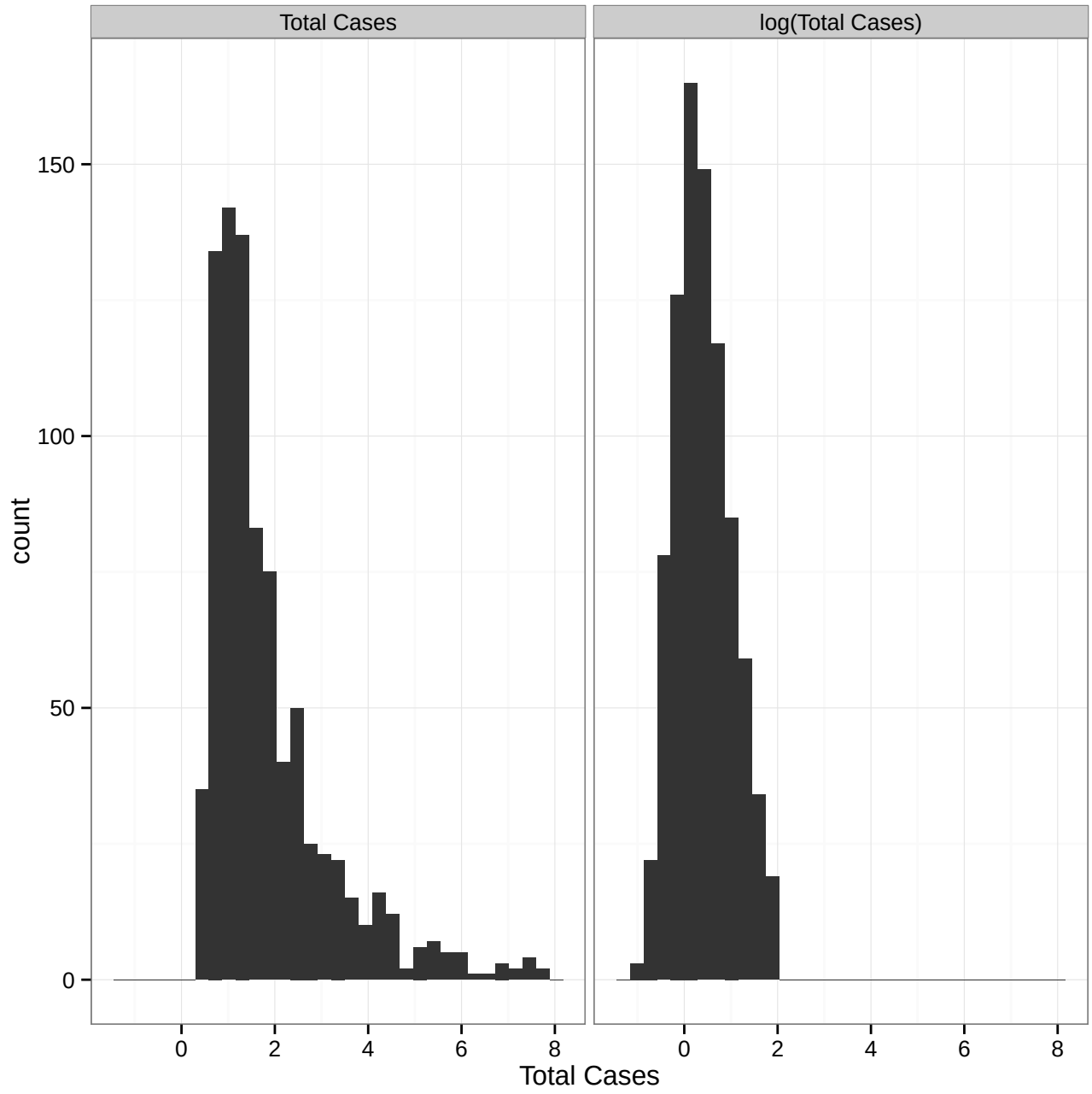
1. Impute the missing values. They are all in the low season, so this should be relatively easy to do.
2. Drop all data up through the last NA.
3. Use the data that are available.

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.

```r
hist_df <- rbind(
        data.frame(value = ili_national$total_cases,
        variable = "Total Cases"),
    data.frame(value = log(ili_national$total_cases),
        variable = "log(Total Cases)")
)

ggplot(aes(x = value), data = hist_df) +
    geom_histogram() +
    facet_wrap( ~ variable, ncol = 2) +
    xlab("Total Cases") +
    theme_bw()
```

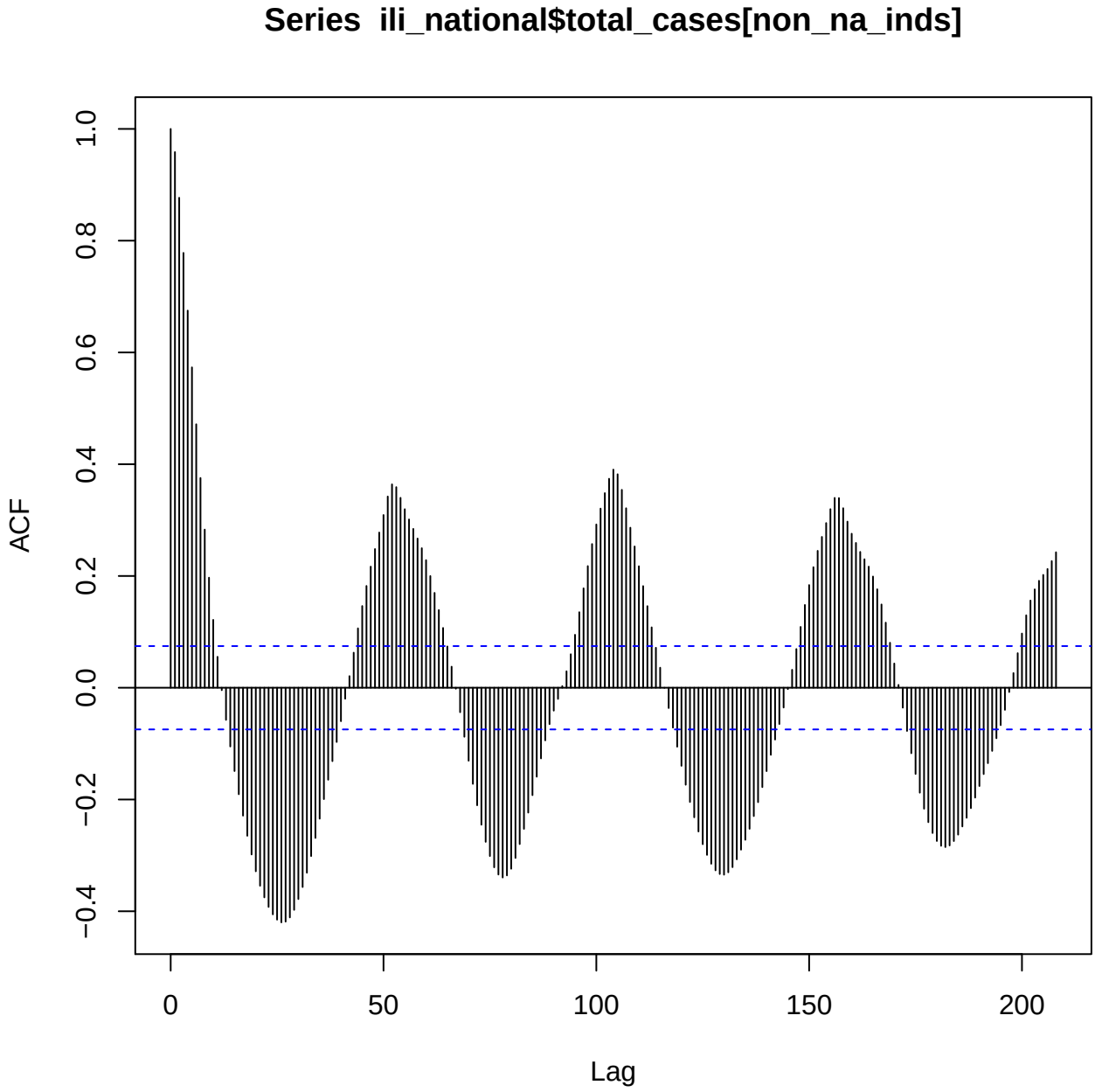
```
## stat_bin:  binwidth defaulted to range/30.  Use 'binwidth = x' to adjust this.
## stat_bin:  binwidth defaulted to range/30.  Use 'binwidth = x' to adjust this.
```



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:

```
last_na_ind <- max(which(is.na(ili_national$total_cases)))
non_na_inds <- seq(from = last_na_ind + 1, to=nrow(ili_national))
acf(ili_national$total_cases[non_na_inds],
  lag.max = 52 * 4)
```



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.

```
## Definitions of kernel components.  A couple of notes:
##   1) In the current implementation, it is required that separate kernel
##      components be used for lagged (predictive) variables and for leading
##      (prediction target) variables.
##   2) The current syntax is verbose; in a future version of the package,
##      convenience functions may be provided.
```

```r
## Define kernel components -- 3 pieces:
##   1) Periodic kernel acting on time index
##   2) pdtmvn kernel acting on lagged total cases (predictive) -- all continuous
##   3) pdtmvn kernel acting on lead total cases (prediction target) -- all continuous
kernel_components <- list(
    list(
        vars_and_offsets = data.frame(var_name = "time_index",
            offset_value = 0L,
            offset_type = "lag",
            combined_name = "time_index_lag0",
            stringsAsFactors = FALSE),
        kernel_fn = periodic_kernel,
        theta_fixed = list(period=pi / 52.2),
        theta_est = list("bw"),
        initialize_kernel_params_fn = initialize_params_periodic_kernel,
        initialize_kernel_params_args = NULL,
        vectorize_kernel_params_fn = vectorize_params_periodic_kernel,
        vectorize_kernel_params_args = NULL,
        update_theta_from_vectorized_theta_est_fn = update_theta_from_vectorized_theta_est_periodic_kernel
        update_theta_from_vectorized_theta_est_args = NULL
    ),
    list(
        vars_and_offsets = data.frame(var_name = "total_cases",
            offset_value = 1L,
            offset_type = "horizon",
            combined_name = "total_cases_horizon1",
            stringsAsFactors = FALSE),
        kernel_fn = pdtmvn_kernel,
        rkernel_fn = rpdtmvn_kernel,
        theta_fixed = list(
            parameterization = "bw-diagonalized-est-eigenvalues",
            continuous_vars = "total_cases_horizon1",
            discrete_vars = NULL,
            discrete_var_range_fns = NULL,
            lower = -Inf,
            upper = Inf
        ),
        theta_est = list("bw"),
        initialize_kernel_params_fn = initialize_params_pdtmvn_kernel,
        initialize_kernel_params_args = NULL,
        vectorize_kernel_params_fn = vectorize_params_pdtmvn_kernel,
        vectorize_kernel_params_args = NULL,
        update_theta_from_vectorized_theta_est_fn = update_theta_from_vectorized_theta_est_pdtmvn_kernel,
        update_theta_from_vectorized_theta_est_args = NULL
    ))
#,
#    list(
#        vars_and_lags = vars_and_lags[3:5, ],
#        kernel_fn = pdtmvn_kernel,
#        rkernel_fn = rpdtmvn_kernel,
#        theta_fixed = NULL,
#        theta_est = list("bw"),
#        initialize_kernel_params_fn = initialize_params_pdtmvn_kernel,
```

```
#         initialize_kernel_params_args = list(
#             continuous_vars = vars_and_lags$combined_name[3:4],
#             discrete_vars = vars_and_lags$combined_name[5],
#             discrete_var_range_fns = list(
#                 c_lag2 = list(a = pdtmvn::floor_x_minus_1, b = floor, in_range = pdtmvn::equals_integer,
#         ),
#         vectorize_theta_est_fn = vectorize_params_pdtmvn_kernel,
#         vectorize_theta_est_args = NULL,
#         update_theta_from_vectorized_theta_est_fn = update_theta_from_vectorized_theta_est_pdtmvn_kernel,
#         update_theta_from_vectorized_theta_est_args = list(
#             parameterization = "bw-diagonalized-est-eigenvalues"
#         )
#     ))
```

Next, we create a list with parameters controlling how estimation is performed.

```
kcde_control <- create_kcde_control(X_names = "time_index",
    y_names = "total_cases",
    time_name = "time",
    prediction_horizons = 1L,
    kernel_components = kernel_components,
    crossval_buffer = ymd("2010-01-01") - ymd("2009-01-01"),
    loss_fn = neg_log_score_loss,
    loss_fn_prediction_type = "distribution",
    loss_args = NULL)
```

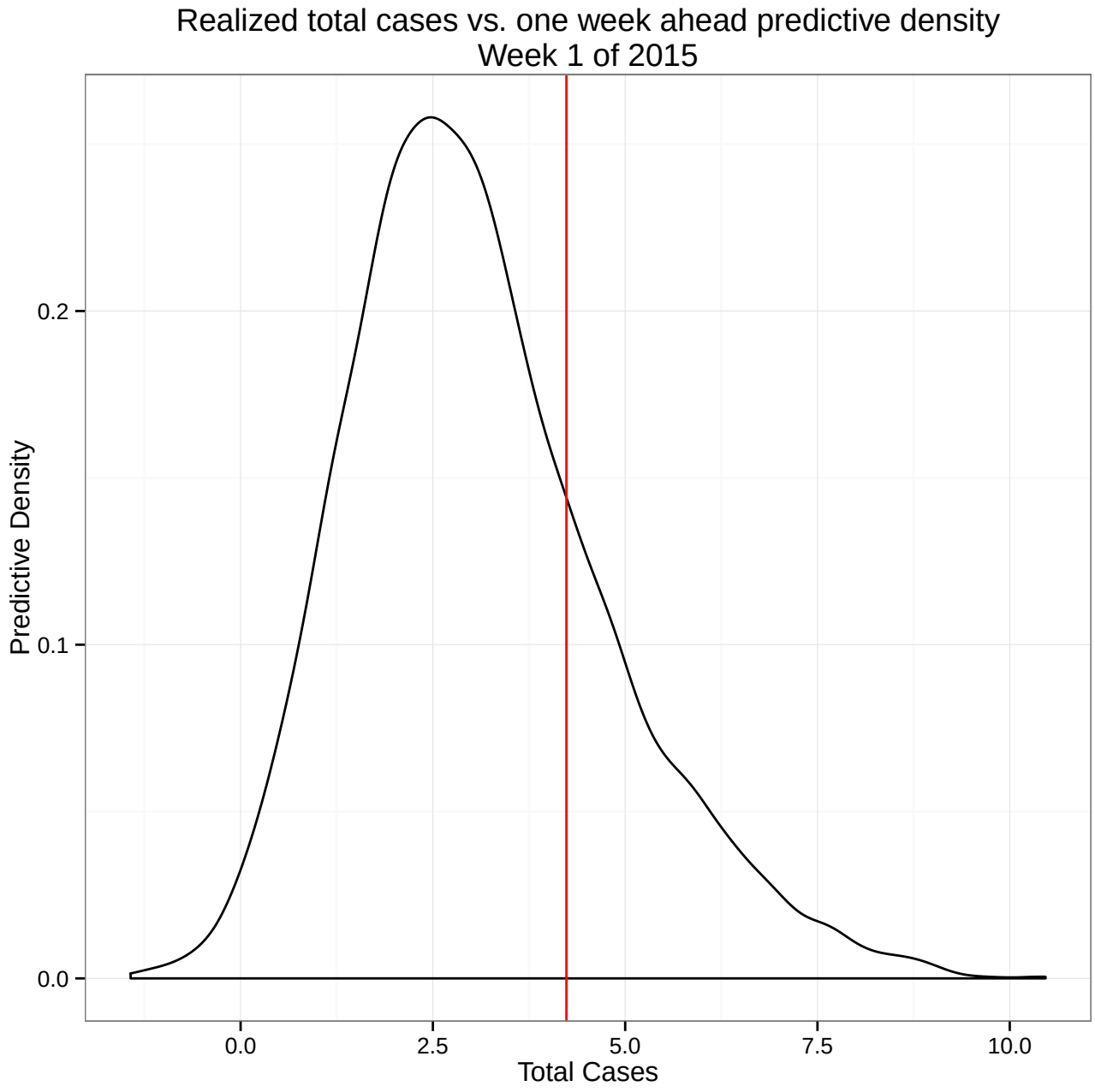
We are now ready to estimate the bandwidth parameters, using data up through 2014

```
flu_kcde_fit_orig_scale <- kcde(data = ili_national[ili_national$year <= 2014, ],
    kcde_control = kcde_control)

## Warning in FUN(X[[i]], ...):  failed to assign RegisteredNativeSymbol for logspace_add_C to logspace_add
since logspace_add_C is already defined in the 'pdtmvn' namespace
## Warning in FUN(X[[i]], ...):  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 'pdtmvn' namespace
## Warning in FUN(X[[i]], ...):  failed to assign RegisteredNativeSymbol for logspace_sub_C to logspace_sub
since logspace_sub_C is already defined in the 'pdtmvn' namespace
## Warning in FUN(X[[i]], ...):  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 'pdtmvn' namespace
```

There are several methods available for examining the predictive distribution. Here we simply draw a monte carlo sample from the predictive distribution for total cases in the first week of 2015. Then we plot a representation of this sample against the observed value for that week.

```
predictive_sample <- kcde_predict(kcde_fit = flu_kcde_fit_orig_scale,
        prediction_data = ili_national[ili_national$year == 2014 & ili_national$week == 53, , drop = FALSE
        leading_rows_to_drop = 0,
        trailing_rows_to_drop = 1L,
        additional_training_rows_to_drop = NULL,
        prediction_type = "sample",
        n = 10000L)

ggplot() +
        geom_density(aes(x = predictive_sample)) +
```

```
        geom_vline(aes(xintercept = ili_national$total_cases[ili_national$year == 2015 & ili_national$week
                colour = "red") +
        xlab("Total Cases") +
        ylab("Predictive Density") +
        ggtitle("Realized total cases vs. one week ahead predictive density\nWeek 1 of 2015") +
        theme_bw()
```



5 Publications

[Byrd et al.(1995)Byrd, Lu, Nocedal, and Zhu] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):

1190–1208, 1995.

6 Examples