Forecasting COVID cases

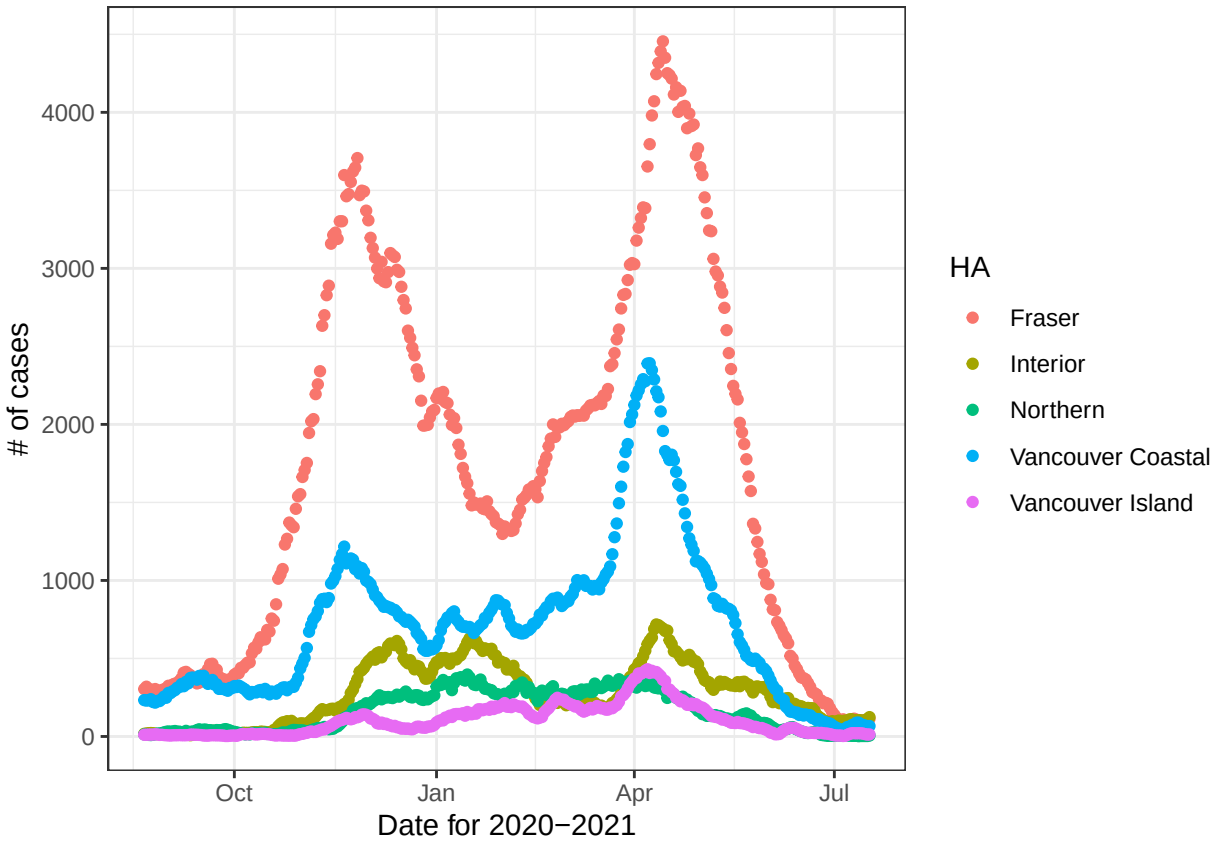
[didemch]

2024-02-01

I will explore several models to forecast COVID-19 cases. The objective is to predict case counts two weeks ahead for all five health authorities. I have two data sets in my repo, `bc-covid-train` and `bc-covid-test`. Both have 6 variables: HA, date, cases+14, cases+0, cases-7, cases-14. `cases+14` is the response: it is the total number of cases observed in the week ending 14 days after `date`. The other `cases` variables are features: the total number of cases in the week ending on the date, 1 week earlier and 2 weeks earlier.

1. Creating a plot displaying the training data of today's cases over time, with points color-coded by health authority.

```
Sys.setenv(LANG = "en")
train <- readRDS("bc-covid-train.RDS")
train$HA = as.factor(train$HA)
ggplot(data=train, aes(y=`cases+0`, x=date, colour = HA))+geom_point()+
xlab("Date for 2020-2021") +
ylab("# of cases")
```



Comment: According to the plot above, we can see that the Fraser health authority had the largest number of Covid cases compared to other heath authorities areas. This Fraser region has peaks in around December 2020 and May 2021. It seems like the Vancouver Island region has the smallest number of cases; and the cases trend doesn't seem to fluctuate as much as it fluctuates in Fraser region.Interestingly, across all health authorities, cases seem to fluctuate, so it is possible to see two "waves": one around Dec 2020 and another one around May 2021. Overall, Fraser and Vancouver Coastal regions have the largest peaks and largest numbers of covid cases.

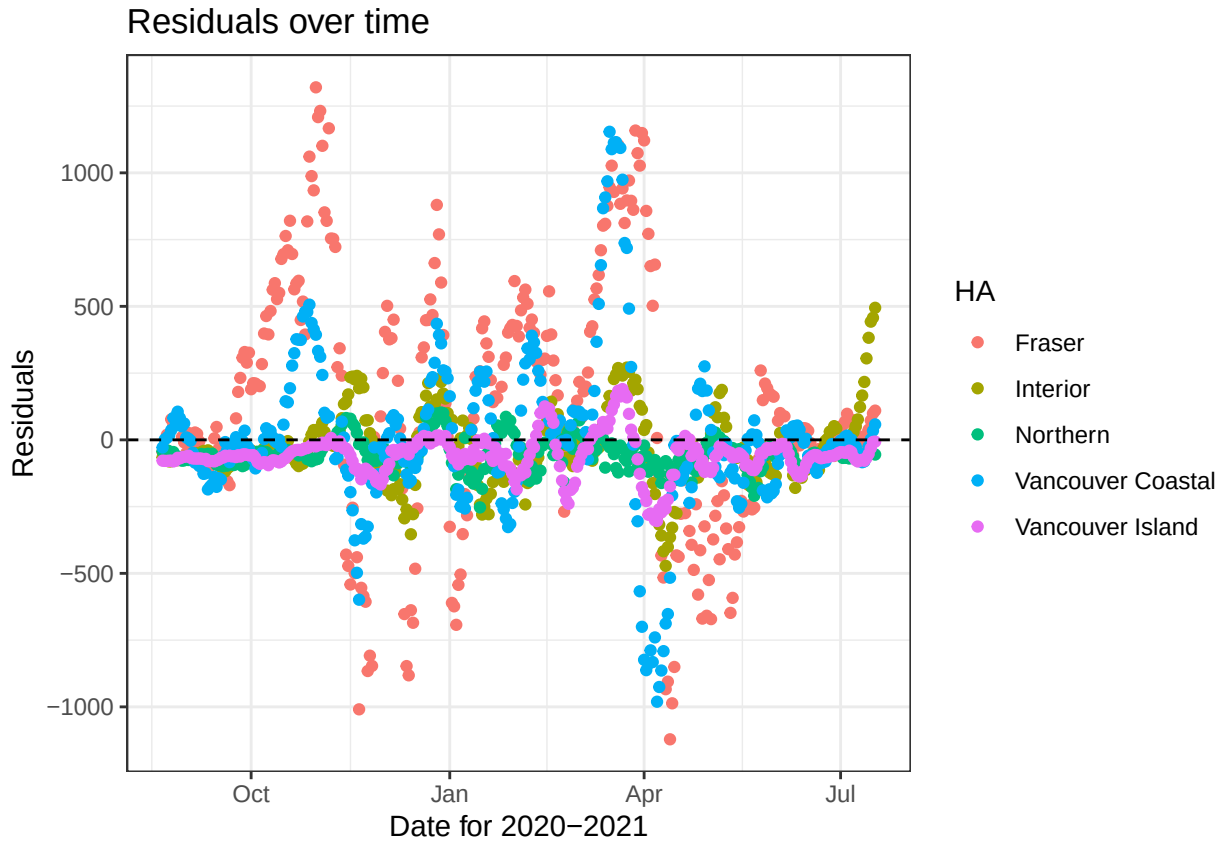
2. Fitting a linear model to predict cases two weeks ahead based on three features using the training data.

```
train$HA = as.factor(train$HA)
full_model <- lm(`cases+14`~`cases+0`+`cases-7`+`cases-14`, data= train)
summary(full_model)
```

```
##
## Call:
## lm(formula = `cases+14` ~ `cases+0` + `cases-7` + `cases-14`,
##     data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1121.48   -90.08   -43.57    42.05  1320.16
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 70.78623    8.19141   8.642   <2e-16 ***
```

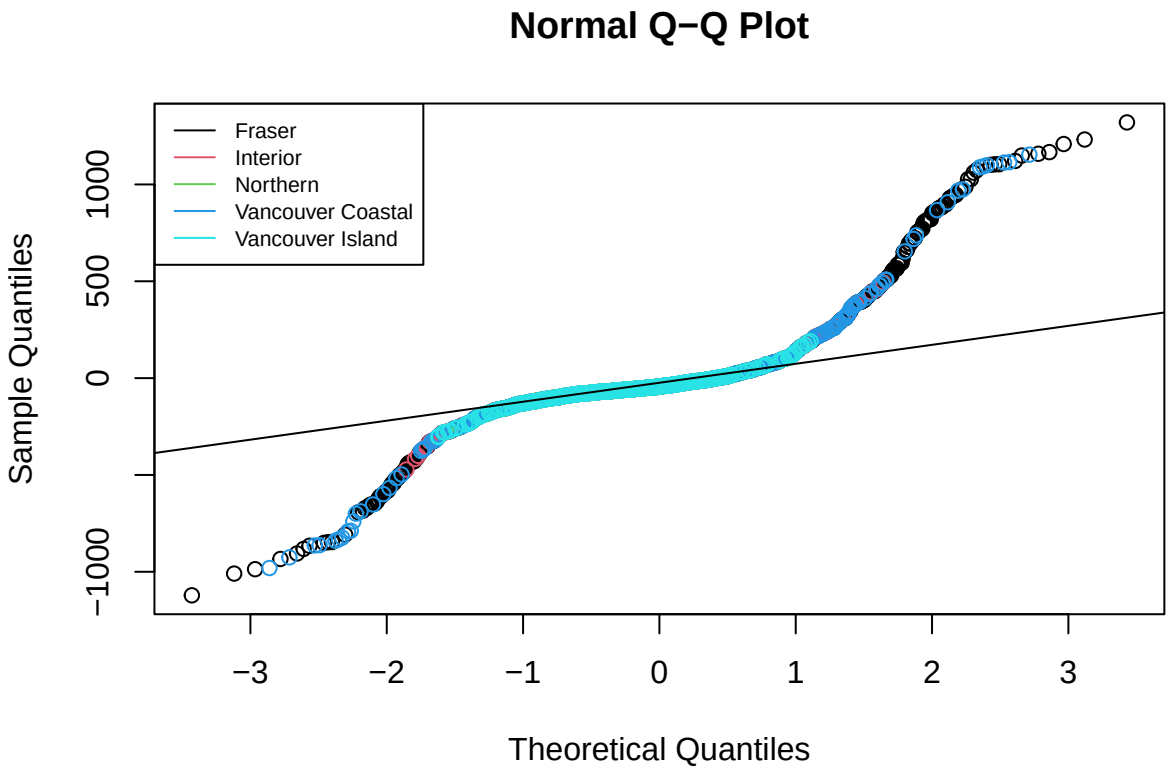
```
## `cases+0`    1.91361    0.04540  42.154   <2e-16 ***
## `cases-7`   -0.92203    0.08066 -11.431   <2e-16 ***
## `cases-14`  -0.10519    0.04536  -2.319   0.0205 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 269.4 on 1651 degrees of freedom
## Multiple R-squared:  0.9026, Adjusted R-squared:  0.9024
## F-statistic:  5098 on 3 and 1651 DF,  p-value: < 2.2e-16
```

```r
#Residuals plot
res_data <- residuals(full_model)
ggplot(data=train, aes(y=res_data, x=date, colour = HA))+geom_point()+
  ggtitle("Residuals over time")+
xlab("Date for 2020-2021") +
ylab("Residuals")+
  geom_hline(yintercept=0,linetype="dashed")
```




```r
# QQ plot by color
#modfit_stdres <- rstandard(full_model)
qqnorm(res_data, main = "Normal Q-Q Plot",
       xlab = "Theoretical Quantiles", ylab = "Sample Quantiles",
       col=train$HA)
qqline(res_data)
legend("topleft", legend = c("Fraser", "Interior", "Northern",
```

```
                                "Vancouver Coastal", "Vancouver Island"),
lty = 1, col =1:length(train$HA), cex = 0.7)
```




```
# Calculating leave-one-out CV.
loocv <- function (mod) {
  mean((residuals(mod))^2 /(1-hatvalues(mod))^2)
}
loocv_value = loocv(full_model)
loocv_value
```

```
## [1] 73396.72
```

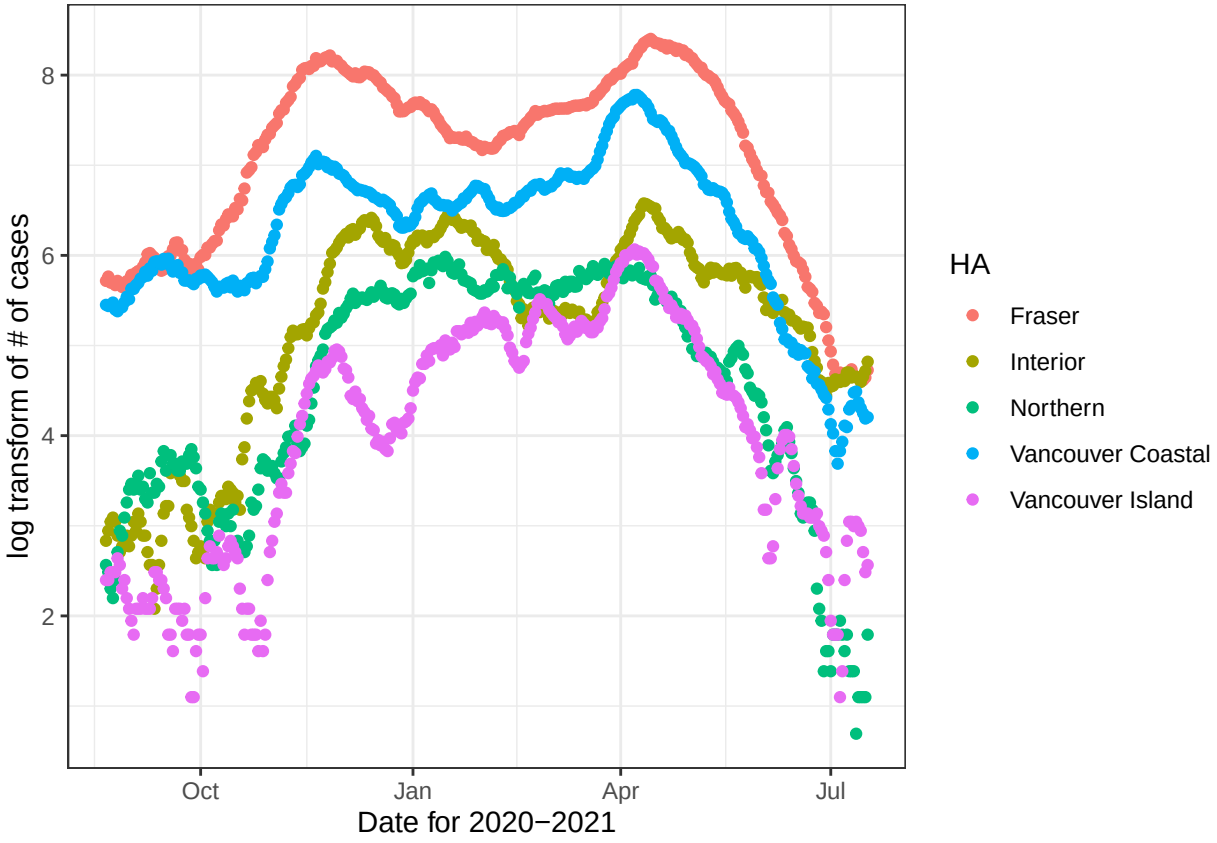
Do these plots seem reasonable? According to the residuals plot, residuals seem to not be symmetrically distributed, tending to cluster towards the middle of the plot. Although most of them are clustered around the lower single digits of the y-axis, there are quite many residual values that are too large. Finally, there seem to be some patterns that residuals follow. Moreover, there are few outliers present. Looking at the QQ-plot, residuals do not follow a straight line; The points seem to fall about a straight line. Residuals do not seem to be normally distributed.

Providing the slope coefficient for `cases+0` and explaining its significance in the context of this problem. Assessing whether this interpretation aligns logically, and providing justification for the assessment The slope coefficient on `cases+0` is estimated to be 1.91361. This means that every time the number of cases in `cases+0` increases by 1, the output number of cases in `cases+14` will increase by 1.91361 cases. This makes sense because if the number of covid cases increases today, in 14 days there will more cases around as the rate of covid spread will increase.

Does this model seem appropriate? The full model does not seem to be a good fit. Although we have nothing to compare our loocv value to, it is possible to see that the value is too large (since ideally we want it to be less than 1). Hence, in the next part we should try to improve the model.

3. Creating a logarithmic scale plot of today's cases over time to compare with the plot in Question 1. Repeating Question 2 with a log transformation of both features and response, using natural logarithm (base e). Additionally, incorporating `HA` as a categorical factor in the model.

```
train$HA = as.factor(train$HA)
ggplot(data=train, aes(y=log(`cases+0`), x=date, colour = HA))+geom_point()+
xlab("Date for 2020-2021") +
ylab("log transform of # of cases")
```



The log transform of both explanatory and response variables was used to approximately conform to normality. Comparing to the plot in Q1, this plot seems to show the differences in fluctuations between regions rather than the difference in # of cases like it was shown in Q1. According to this plot, Vancouver Island had the largest difference between minimal and maximal number of cases in 2020-2021. From plot in Q1 I was not able to detect this.

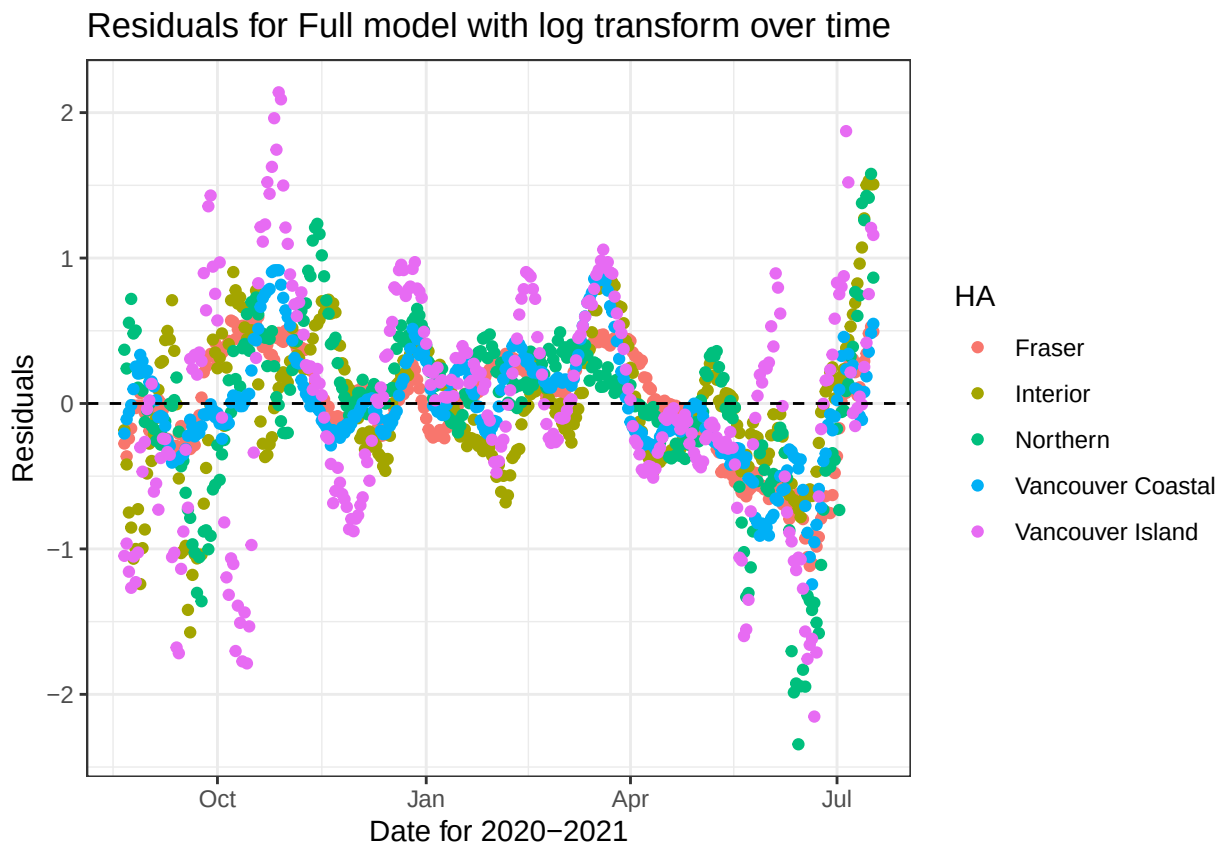
Repeating Question 2 with log transforms.

```
train$HA = as.factor(train$HA)
full_model_log <- lm(log(`cases+14`)~log(`cases+0`)+log(`cases-7`)+
                       log(`cases-14`)+HA, data= train)
summary(full_model_log)
```

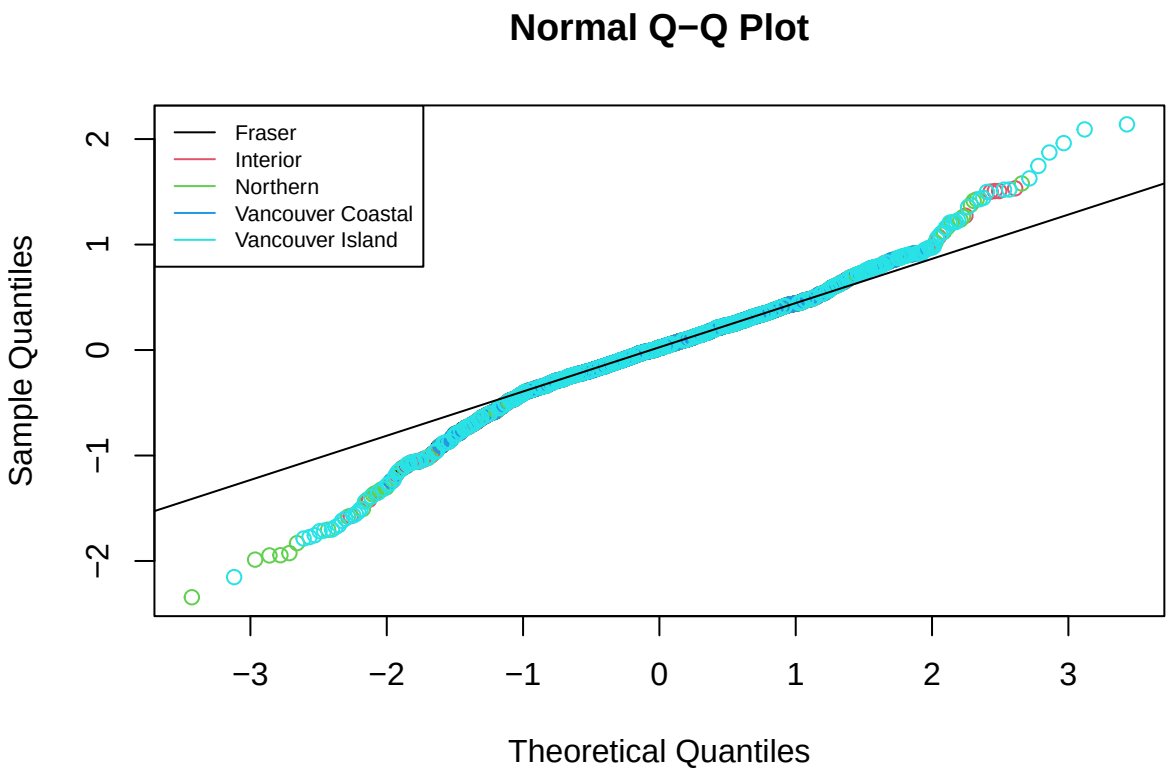
```
##
```

```
## Call:
## lm(formula = log(`cases+14`) ~ log(`cases+0`) + log(`cases-7`) +
##     log(`cases-14`) + HA, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.34362 -0.25656  0.02116  0.30902  2.13952
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1.22168    0.08921  13.694  < 2e-16 ***
## log(`cases+0`)        1.16077    0.03433  33.807  < 2e-16 ***
## log(`cases-7`)       -0.02462    0.04885  -0.504 0.614321
## log(`cases-14`)      -0.31004    0.03407  -9.099  < 2e-16 ***
## HAInterior           -0.23058    0.04777  -4.826 1.52e-06 ***
## HANorthern           -0.44159    0.05153  -8.570  < 2e-16 ***
## HAVancouver Coastal -0.15241    0.04253  -3.583 0.000349 ***
## HAVancouver Island  -0.49312    0.05556  -8.875  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5328 on 1647 degrees of freedom
## Multiple R-squared:  0.8885, Adjusted R-squared:  0.888
## F-statistic:  1874 on 7 and 1647 DF,  p-value: < 2.2e-16
```

```
#Residuals plot
res_data_log <- residuals(full_model_log)
ggplot(data=train, aes(y=res_data_log, x=date, colour = HA))+geom_point()+
  ggtitle("Residuals for Full model with log transform over time")+
xlab("Date for 2020-2021") +
ylab("Residuals")+
  geom_hline(yintercept=0,linetype="dashed")
```




```r
# QQ plot by color
qqnorm(res_data_log, main = "Normal Q-Q Plot",
       xlab = "Theoretical Quantiles", ylab = "Sample Quantiles",
       col=train$HA)
qqline(res_data_log)
legend("topleft", legend = c("Fraser", "Interior", "Northern",
                             "Vancouver Coastal", "Vancouver Island"),
lty = 1, col =1:length(train$HA), cex = 0.7)
```




```
# Calculating leave-one-out CV.
loocv <- function (mod) {
  mean((residuals(mod))^2 /(1-hatvalues(mod))^2)
}

loocv_value_log = loocv(full_model_log)
loocv_value_log
```

```
## [1] 0.2859956
```

Do these plots seem reasonable? The QQ-plot now has a clearer pattern, which resembles a straight line. However, the plot includes skewed ends and outliers. Overall, it seems like the QQ-plot was improved by applying the log-transformation. The residuals plot also seems to be improved since the clear pattern now became harder to see in the first part of the plot from Sept 2020 to around Non 2020. What is more, now residuals for Vancouver Island and Northern regions are more spread out as opposed to being clustered around 0 as in the previous plot.

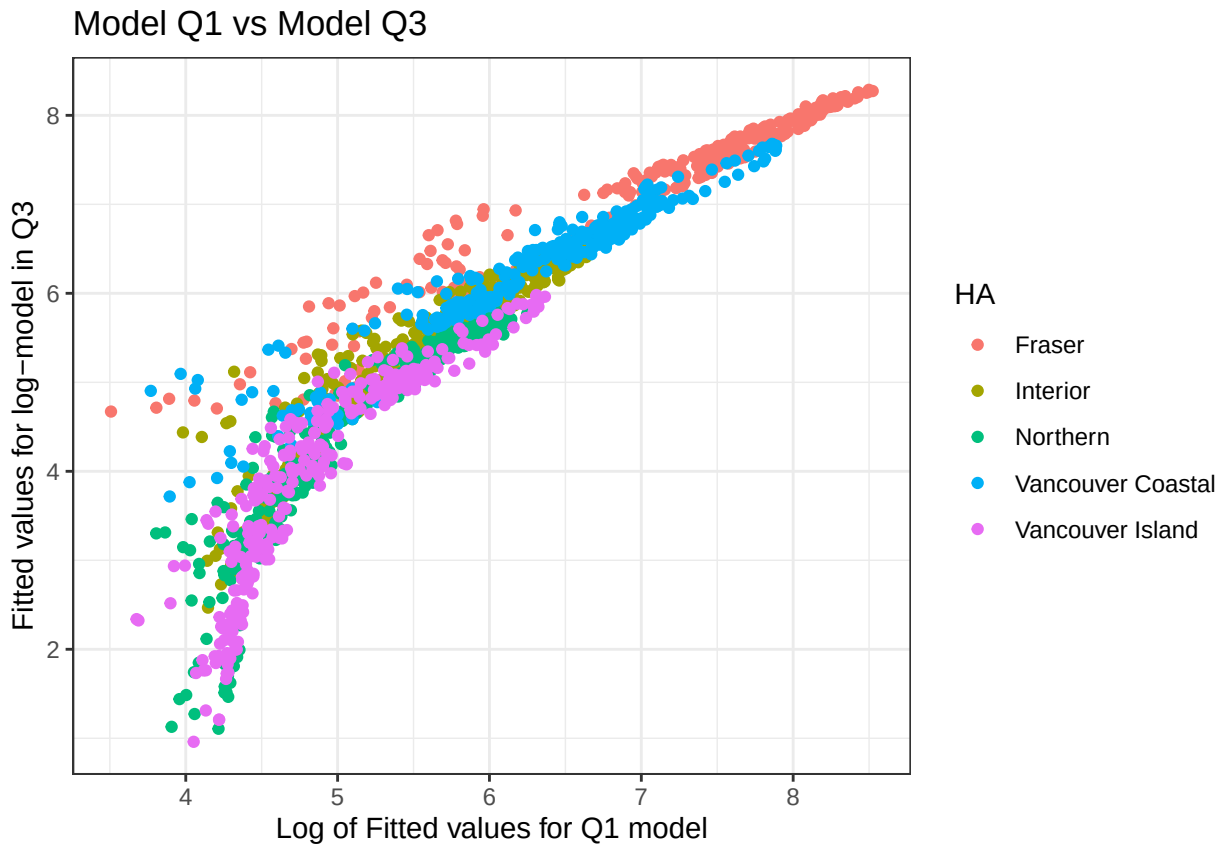
Providing the slope coefficient for `cases+0` *and explaining its significance in the context of this problem. Assessing whether this interpretation aligns logically, and providing justification for the assessment.* The slope coefficient on log(`cases+0`) is estimated to be 1.16077. After the log-transformation of both response and explanatory variables, the interpretation becomes quite difficult. There is a trade-off between model quality and model interpretation. However, there is still a positive relationship between the log(`cases+0`) and log(`cases+14`) since the coefficient in front of log(`cases+0`) is positive. Hence, if the log(`cases+0`) increases, the log(`cases+14`) should also increase.

Does this model seem appropriate? Since the value of loocv is 0.286, which is less than 1, it seems like

this model is much better than our first model. MSPE is much lower in this case. Hence, this model is appropriate, however, it is more difficult to interpret.

4. Plotting the fitted values from the two models against each other, making sure to transform them as needed to be comparable. (Coloring the points by HA).

```
train$HA = as.factor(train$HA)
ggplot(data=train, aes(y=full_model_log$fitted.values,
                       x=log(full_model$fitted.values), colour = HA))+
  geom_point()+
xlab("Log of Fitted values for Q1 model") +
ylab("Fitted values for log-model in Q3") +
  ggtitle("Model Q1 vs Model Q3")
```




```
## Calculating the mean absolute error and the root mean squared error
#of both models.

mae_Q1 = mae(train$`cases+14`, full_model$fitted.values)
cat("Mean absolute error for Q1 model: ", mae_Q1, sep="\n")
```

```
## Mean absolute error for Q1 model:
## 160.8978
```

```r
mae_Q3 = mae(log(train$`cases+14`), full_model_log$fitted.values)
cat("Mean absolute error for Q3 model: ", mae_Q3, sep="\n")
```

```
## Mean absolute error for Q3 model:
## 0.3886077
```

```r
rmse_Q1 =rmse(train$`cases+14`, full_model$fitted.values)
cat("Root mean squared error for Q1 model: ", rmse_Q1, sep="\n")
```

```
## Root mean squared error for Q1 model:
## 269.0613
```

```r
rmse_Q3 = rmse(log(train$`cases+14`), full_model_log$fitted.values)
cat("Root mean squared error for Q3 model: ", rmse_Q3, sep="\n")
```

```
## Root mean squared error for Q3 model:
## 0.5314758
```

Discussing the results: Looking at the Model Q1 vs Model Q3 plot above, it seems like they have a log pattern. It does not seem that fitted values in Q1 model are the same as fitted values in Q3 model. But this is expected since it was declared that model in Q3 was a better fit. Since model with log-transformation has much lower mean absolute error and root mean squared error than model in Q1, it once again supports the claim about a good fit of model Q3. Lower values of RMSE and MAE indicate better fit. However, just by looking at the model Q3, its RMSE value is not that low. In my previous STATs classes RMSE had to be between 0.2 and 0.5 in order to support a good model. In this case, RMSE is just slightly above the threshold. What is more, the difference between RMSE and MAE indicates there is some variation in the magnitude of errors. And the average difference between observed and fitted number of log of covid cases was 0.5314758-0.3886077=0.1428681.

5. Can the CV scores calculated in the previous two problems be used to choose between these models? No, it cannot. This is because our calculated loocv values have a very high variance since n=1. As discussed in class, loocv is a slightly pessimistic estimation of the model performance. Hence, rather than choosing a model based on a loocv, it is better to use a cv with k=5 or more.

6. Writing a function that calculates K-fold CV for either model.

```r
#' @param data The full data set
#' @param estimator Function. Has 1 argument (some data) and fits a model.
#' @param predictor Function. Has 2 args (the fitted model, the_newdata)
#and produces predictions
#' @param kfolds Integer. The number of folds.
#' @param responsename Character. The column name of "cases+14" in the data

kfold_cv <- function(data,
                     estimator,
                     predictor,
                     kfolds = 5,
                     responsename = "cases+14") {
  n <- nrow(data)
  fold.labels <- sample(rep(1:kfolds, length.out = n))
  maee <- double(kfolds)
```

```
  for (fold in 1:kfolds) {
    test.rows <- fold.labels == fold
    train <- data[!test.rows,]
    test <- data[test.rows,]
    current_model <- estimator(train)
    predictions <- predictor(current_model, test)
    test_responses <- test[, responsename]
    test_errors <- test_responses - predictions
    absolute_errors <- abs(test_errors)
    maee[fold] <- mean(absolute_errors[, 1])
  }
  mean(maee)
}
```

7. Applying my function to both models with $K = 20$.

```
est_full <- function(dataset) full_model
pred <- function(mod, dataset) predict(full_model, newdata = dataset)
cat("The CV 20-fold value for the full model is ",kfold_cv(train, est_full,
                                    pred, 20, "cases+14"), sep="\n")
```

```
## The CV 20-fold value for the full model is
## 160.919
```

```
train$`cases+14_log`=log(train$`cases+14`)
est_log <- function(dataset) full_model_log
pred_log <- function(mod, dataset) predict(full_model_log, newdata = dataset)
cat("The CV 20-fold value for the full log transformed  model is ",
    kfold_cv(train, est_log, pred_log, 20, "cases+14_log"))
```

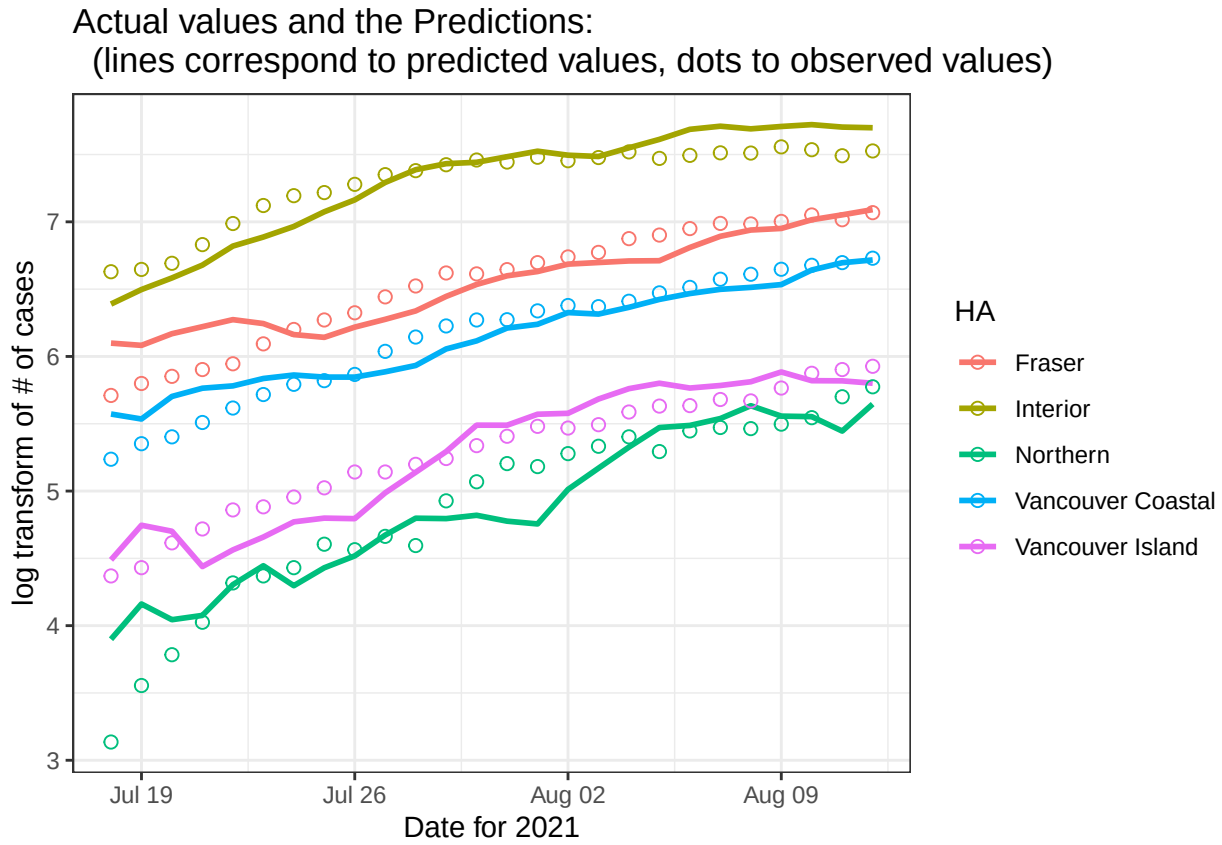
```
## The CV 20-fold value for the full log transformed  model is  0.3884988
```

Since we want the CV value to be smaller because it means smaller errors, it seems like the transformed model from Q3 is a better choice here; it has a smaller CV value compared to the full model. This log-transformed model is more accurate than model in Q1 by 160.8667/0.3887073=413.8505 times.

8. Utilizing my preferred model to predict the test set.

```
test <- readRDS("bc-covid-test.RDS")
full_model_log_test <- lm(log(`cases+14`)~log(`cases+0`)+log(`cases-7`)+
                            log(`cases-14`)+HA, data= test)

predicted_values <- predict(full_model_log_test, data=test)
test$HA = as.factor(test$HA)
ggplot(data=test, aes(y=predicted_values, x=date, colour = HA))+
  geom_line( size=1)+
  geom_point(aes(y =log(`cases+14`), x=date), shape=1, size = 2)+
xlab("Date for 2021") +
ylab("log transform of # of cases")+
  ggtitle("Actual values and the Predictions:
  (lines correspond to predicted values, dots to observed values)")
```



For Northern region, the predicted and fitted values seem to be very far away from each other. It seems like the fitted values do not even follow the fluctuations of log -transformed covid cases in this region. For Vancouver Island, predicted values seem to underestimate and then overestimate the cases. Same happened for the Vancouver Coastal and Fraser regions. For the Interior region, somewhere around end of July 2021 the predicted values approximately equal to the observed values. However, later predicted values start to overestimate the log # of cases.

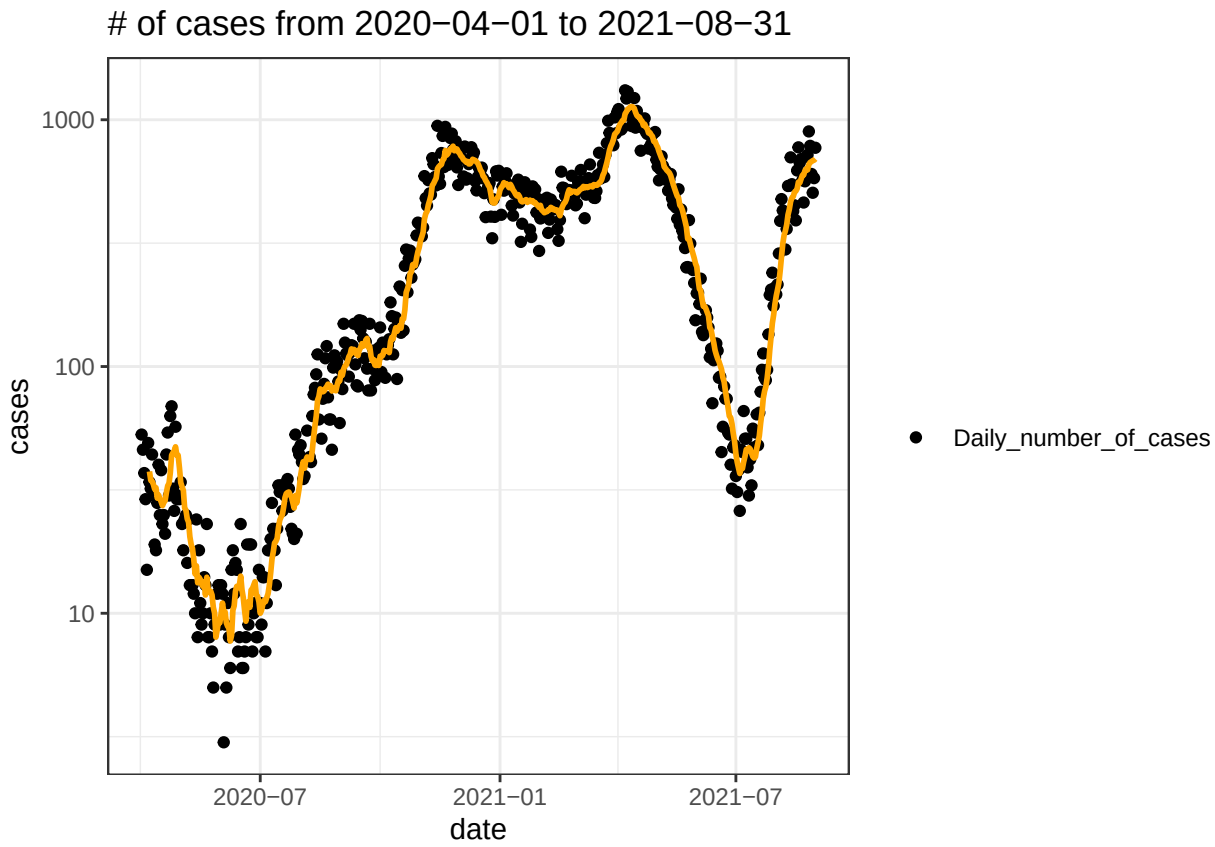
Nonparametric regression

```
bc_cases <- readRDS("bc-covid-cases.RDS")
```

1. Plotting the `bc_cases` data on the log-scale along with a 7-day trailing average:$\overline{y}_i = \frac{1}{7}\sum_{j=i-6}^{i} y_j$ to "smooth" this data. I want to know how well does this smoother "track" the data?

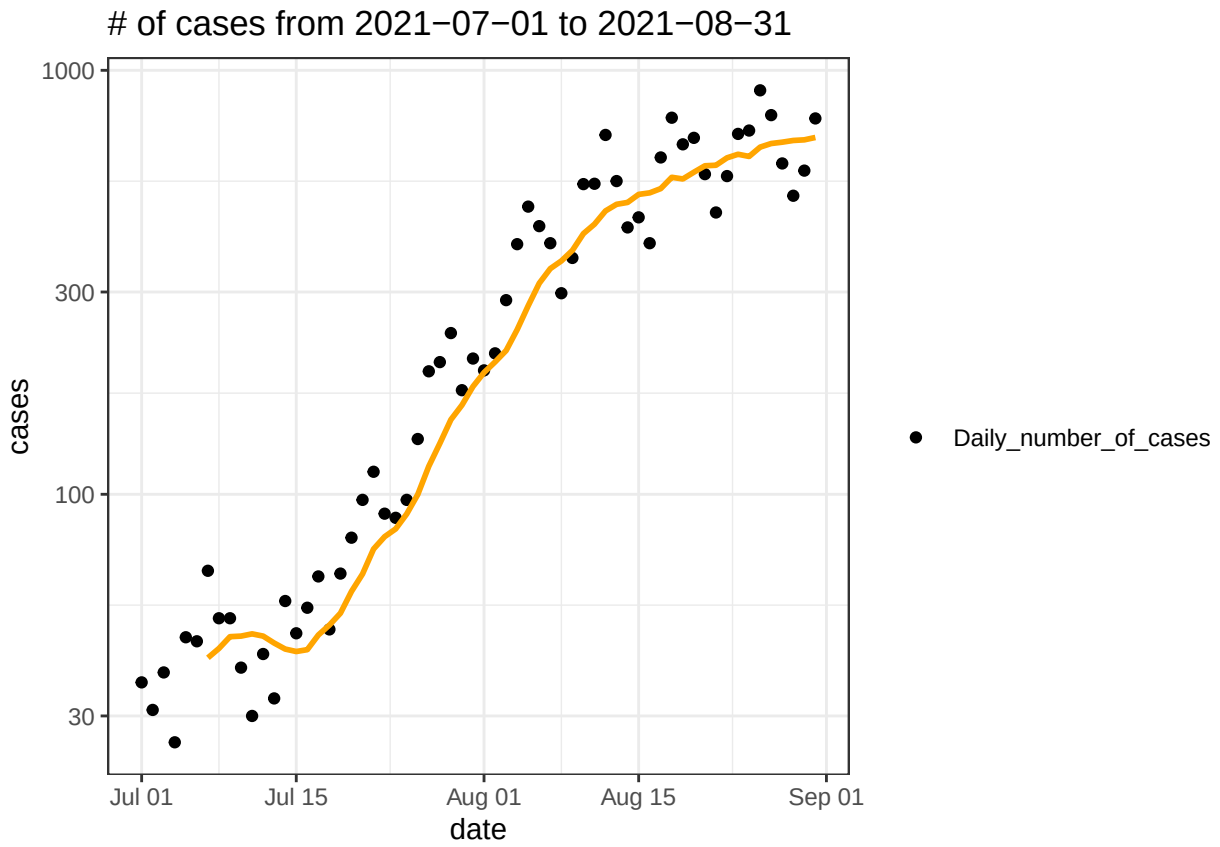
```
library(zoo)
bc_cases_ts <- ts(bc_cases$cases)
weekly_avg_cases <- rollmean(bc_cases_ts, 7, align = "right", fill = NA)

ggplot(bc_cases, aes(date, cases))+geom_point(aes(col="Daily_number_of_cases"))+
  geom_line(aes(x=date, y=weekly_avg_cases), color="orange", size=1) +
  scale_colour_manual("", values= c(Daily_number_of_cases="black",
                                    "Weekly average of cases" = "orange"))+
  scale_y_log10()+ ggtitle("# of cases from 2020-04-01 to 2021-08-31")
```




```
# The most recent 2 months of data
last2mnths <- bc_cases[457:518, ]
last2mnths_ts <- ts(last2mnths$cases)
smooth_last2mnths <- rollmean(last2mnths_ts, 7, align = "right", fill = NA)

ggplot(last2mnths, aes(date, cases))+geom_point(aes(col="Daily_number_of_cases"))+
  geom_line(aes(x=date, y=smooth_last2mnths), color="orange", size=1) +
  scale_colour_manual("", values= c(Daily_number_of_cases="black",
                                    "Weekly average of cases" = "orange"))+
  scale_y_log10()+ ggtitle("# of cases from 2021-07-01 to 2021-08-31")
```



First thing that I see is that the rolling mean does not precisely reach some of the peaks that occur around May-June 2020 or July 2021. This means the smoother is not that good at recording the fluctuations on a short time scale. Overall, during longer periods of time, seems like the rolling mean does a good job of following the data change (from september 2020 to february 2021).
According to the plot of the last two months, the smoother line follows the overall trend of the data. Overall, it is possible to conclude that smooother does a good job in predicting changes that happen over a longer period of time, whereas for short rapid changes the smoother is unable to track them.

2. The 7-day trailing average is a "linear smoother". Completing the function below that creates the smoothing matrix for any dimension `n` and any "window" of days `k`. I assume that the data is equally spaced and arranged in increasing order.

```
#According to the math formula above, here we consider right aligned roll-mean.

trail_mean_mat <- function(n, k = 3) {
  stopifnot(n == floor(n), k == floor(k), n > 0, k > 0) # check for valid inputs
  mat <- matrix(0, nrow = n, ncol = n) # preallocate space
  W = (k - 1) / 2
  for (i in 1:n) {
    # loop over rows
    idx <-   (i - k + 1):(i) # which columns are nonzero?
    denom <- length(idx)
    case1 = i - k
    if (case1 < 0) {
      mat[i,] = NA
    }
```

```
    else {
      mat[i, idx] = 1 / denom
    }
  }
  mat
}

round(trail_mean_mat(8, 3), 2)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]   NA   NA   NA   NA   NA   NA   NA   NA
## [2,]   NA   NA   NA   NA   NA   NA   NA   NA
## [3,] 0.33 0.33 0.33 0.00 0.00 0.00 0.00 0.00
## [4,] 0.00 0.33 0.33 0.33 0.00 0.00 0.00 0.00
## [5,] 0.00 0.00 0.33 0.33 0.33 0.00 0.00 0.00
## [6,] 0.00 0.00 0.00 0.33 0.33 0.33 0.00 0.00
## [7,] 0.00 0.00 0.00 0.00 0.33 0.33 0.33 0.00
## [8,] 0.00 0.00 0.00 0.00 0.00 0.33 0.33 0.33
```

3. What is the effective degrees of freedom of the "trailing average" for $n = 50$ and $k = 10$? For n large relative to k, what does this look like for arbitrary k?

```
library(psych)
mat2 <- trail_mean_mat(50, 10)
tr(mat2)
```

```
## [1] 4.1
```

```
# Examining $lim_{n\rightarrow\infty} edf(n,k) / n$ on example
tr(round(trail_mean_mat(100, 5), 2))
tr(round(trail_mean_mat(500, 5), 2))
tr(round(trail_mean_mat(900, 5), 2))
900/5
tr(round(trail_mean_mat(1000, 5), 2))
tr(round(trail_mean_mat(2900, 5), 2))
tr(round(trail_mean_mat(5000, 5), 2))
500/5
tr(round(trail_mean_mat(15000, 5), 2))
15000/5
```

Effective degrees of freedom is the trace of matrix S (the one found above). Hence, the effective degrees of freedom of the "trailing average" for $n = 50$ and $k = 10$ is 4.1.
According to the example I did above to examine $lim_{n\to\infty} edf(n,k)/n$, it seems like eventually, as n gets relatively large compared to k, the $edf(n,k)/n$ would be approaching n/k.

4. The following function generates the smoothing matrices, regression estimates, and the EDF for Gaussian and Boxcar kernels with different bandwidths.

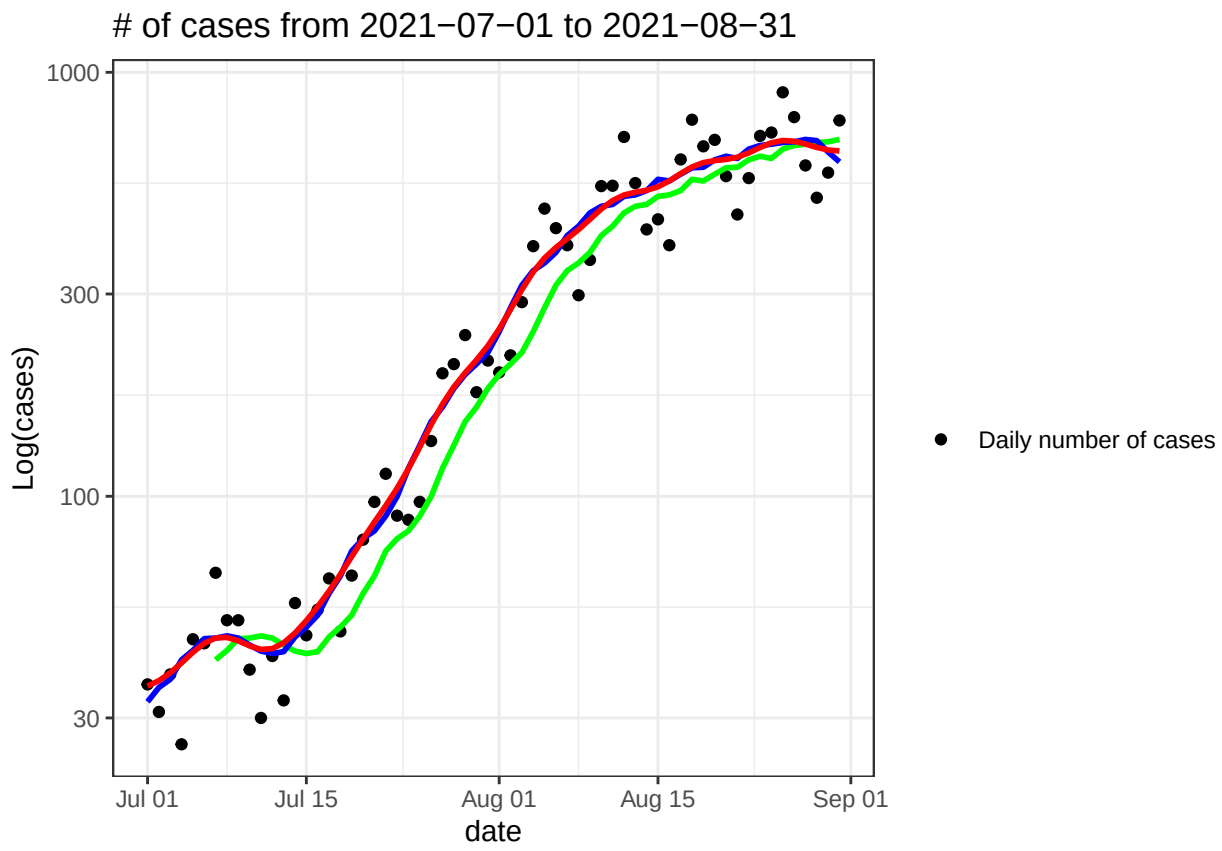
```
kernel_smoother <- function(x, y, kern = c("boxcar", "gaussian"), band = 7) {
  dmat <- as.matrix(dist(x))
  kern <- match.arg(kern)
```

```
  W <- switch(kern,
              boxcar = dmat <= (band * 0.5),
              gaussian = dnorm(dmat, 0, sd = band * 0.3706506))
  W <- sweep(W, 1, rowSums(W), '/')
  fit <- W %*% y
  out <- list(fit = fit, edf = sum(diag(W)))
  out
}
```

Using just the data on or after 1 July 2021. Using the function to plot the data along with (1) the trailing 7-day average, (2) the boxcar smoother with `band = 7` and (3) the Gaussian smoother with `band = 7`. I want to see how results from the two new smoothers compare with those of the 7-day trailing average?

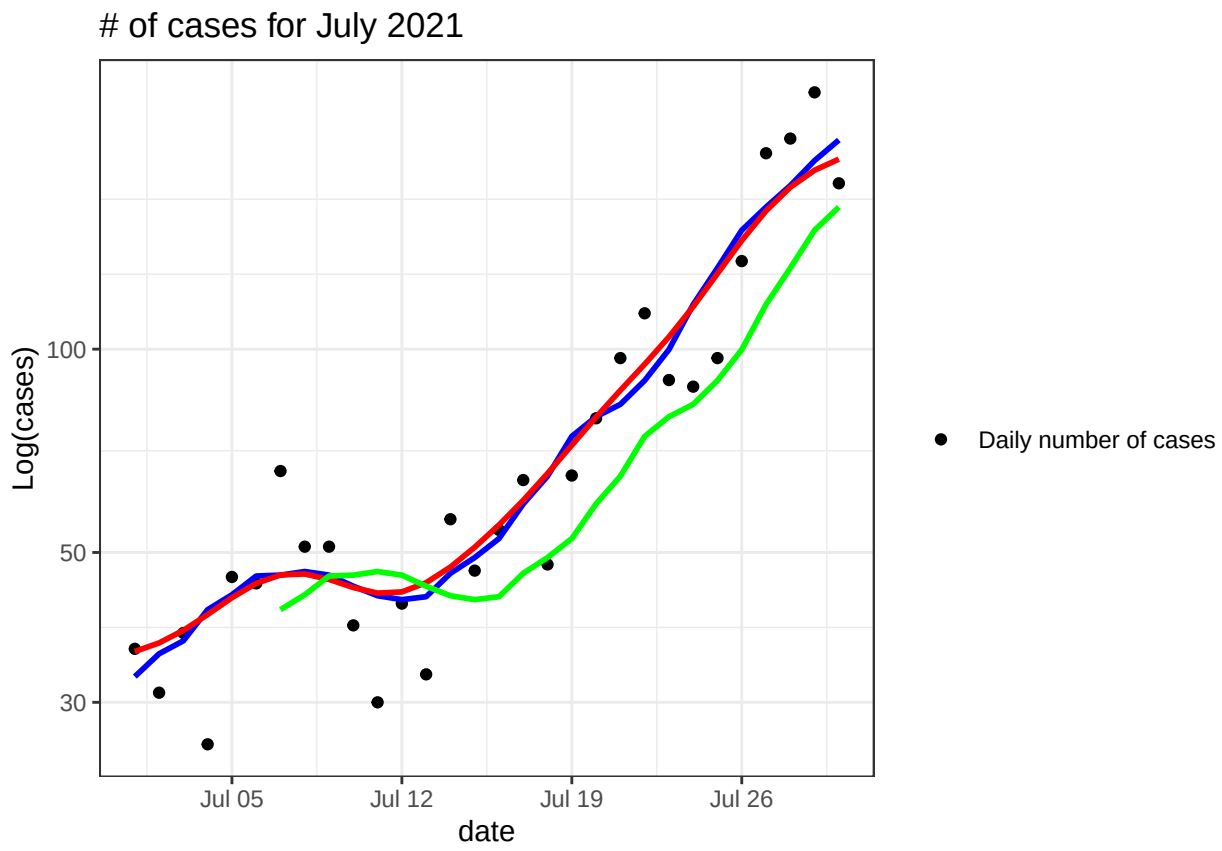
```
bc_smoothed <- bc_cases[457:518, ] #data on or after 1 July 2021
boxcar_smooth <- kernel_smoother(bc_smoothed$date, bc_smoothed$cases,
                                 kern="boxcar", band=7)
gaus_smooth <- kernel_smoother(bc_smoothed$date, bc_smoothed$cases,
                               kern="gaussian", band=7)
last2mnths_ts <- ts(bc_smoothed$cases)
smooth_last2mnths <- rollmean(last2mnths_ts, 7, align = "right", fill = NA)


ggplot(bc_smoothed, aes(date, cases)) +geom_point(aes(col="Daily number of cases")) +
  geom_line(aes(y=smooth_last2mnths), color="green", size=1)+
  geom_line(aes(date, boxcar_smooth$fit), color="blue", size=1)+
  geom_line(aes(date, gaus_smooth$fit), color="red", size=1)+
  scale_colour_manual("", values= c(`Daily number of cases`="black",
                                    "Trailing 7-day average" = "green",
                                    "Boxcar smoother with `band = 7`" = "blue",
                                    "Gaussian smoother with `band = 7`"="red"))+
  scale_y_log10()+ ggtitle("# of cases from 2021-07-01 to 2021-08-31")+
  ylab("Log(cases)")
```



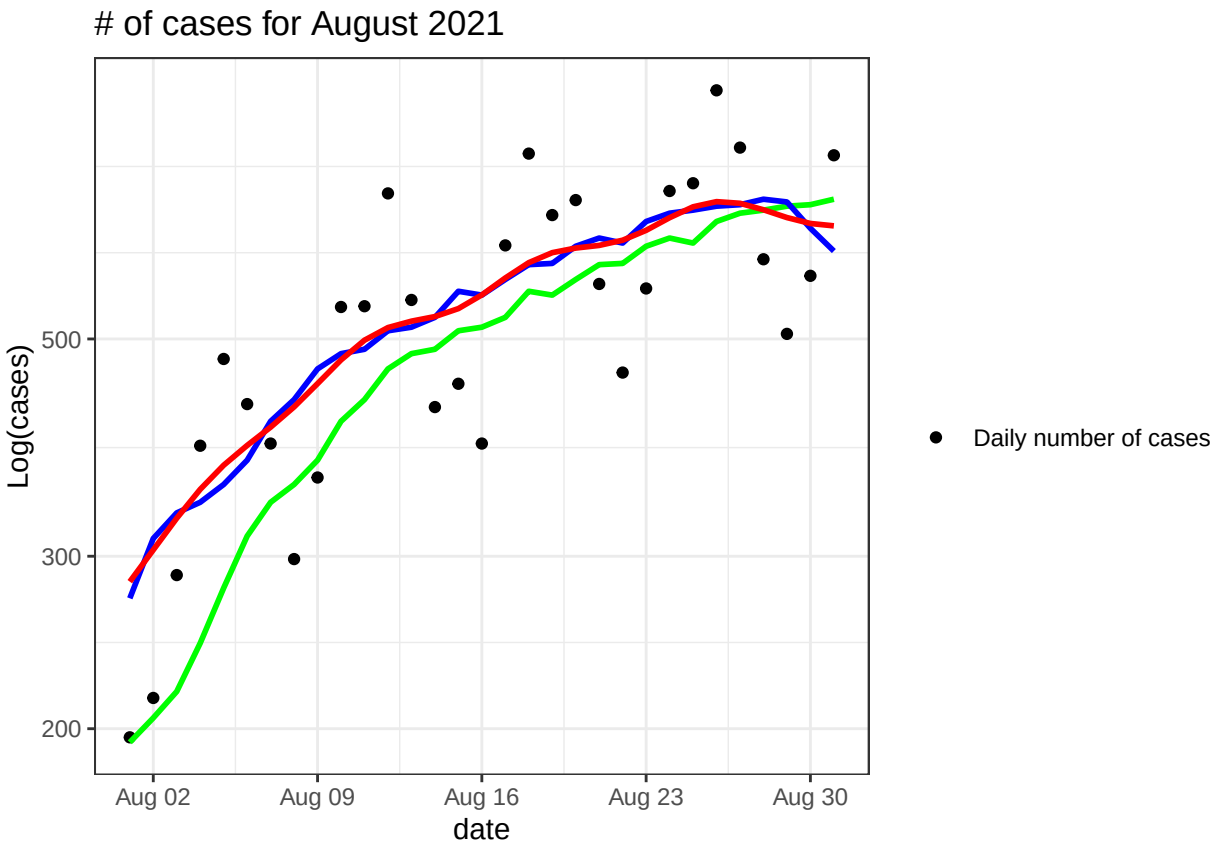

```
# For July 2021
bc_smoothed_july <- bc_smoothed[1:30, ] #data for July 2021
boxcar_smooth <- kernel_smoother(bc_smoothed_july$date, bc_smoothed_july$cases,
                                 kern="boxcar", band=7)
gaus_smooth <- kernel_smoother(bc_smoothed_july$date, bc_smoothed_july$cases,
                               kern="gaussian", band=7)
last2mnths_ts <- ts(bc_smoothed_july$cases)

ggplot(bc_smoothed_july, aes(date, cases)) +
  geom_point(aes(col="Daily number of cases")) +
  geom_line(aes(date, boxcar_smooth$fit), color="blue", size=1)+
  geom_line(aes(date, gaus_smooth$fit), color="red", size=1)+
  geom_line(aes(y=smooth_last2mnths[1:30]), color="green", size=1)+
  scale_colour_manual("", values= c(`Daily number of cases`="black",
                                    "Trailing 7-day average" = "green",
                                    "Boxcar smoother with `band = 7`" = "blue",
                                    "Gaussian smoother with `band = 7`"="red"))+
  scale_y_log10()+ ggtitle("# of cases for July 2021")+
  ylab("Log(cases)")
```




```
# For August 2021
bc_smoothed_aug <- bc_smoothed[32:62, ] #data for July 2021
boxcar_smooth <- kernel_smoother(bc_smoothed_aug$date,
                                 bc_smoothed_aug$cases, kern="boxcar", band=7)
gaus_smooth <- kernel_smoother(bc_smoothed_aug$date,
                               bc_smoothed_aug$cases, kern="gaussian", band=7)
last2mnths_ts <- ts(bc_smoothed_aug$cases)

ggplot(bc_smoothed_aug, aes(date, cases)) +
  geom_point(aes(col="Daily number of cases")) +
  geom_line(aes(y=smooth_last2mnths[32:62]), color="green", size=1)+
  geom_line(aes(date, boxcar_smooth$fit), color="blue", size=1)+
  geom_line(aes(date, gaus_smooth$fit), color="red", size=1)+
  scale_colour_manual("", values= c(`Daily number of cases`="black",
                                    "Trailing 7-day average" = "green",
                                    "Boxcar smoother with `band = 7`" = "blue",
                                    "Gaussian smoother with `band = 7`"="red"))+
  scale_y_log10()+ ggtitle("# of cases for August 2021")+
  ylab("Log(cases)")
```



How do the results from the two new smoothers compare with those of the 7-day trailing average?

I used allign center for the rolling mean since it is given in this problem. According to the plot above, new smoothers overall follow the similar pattern. However, the trailing average seems to be very off to the right. If we look closely at July 2021, then the Gaussian smoother with `band = 7` seems to be the smoothest. Trailing 7-day average and Boxcar smoother with `band=7` seems to be affected by local values more. Moreover, the trailing average seems to be off for July 2021 as compared to the other two smooothers. Same happens in August 2021 too. The Boxcar seems to be more affected by local values rather than the Gaussian. This is because Gaussian considers all of the points but gives more weight to points around. Boxcar considers only k(width of band) values around. Again, the trailing average differs by a lot from the other two.

5. Adjusting the `kernel_smoother()` function so that it also computes and returns the LOO-CV score. Computing the the LOO-CV score for each integer value of `band` from 1 to 21 for the Gaussian kernel. Ploting the scores against `band`. Which value is best? Will the resulting plot be smoother or wigglier than with `band = 7`?

```
kernel_smoother <-
  function(x, y, kern = c("boxcar", "gaussian"), band) {
    dmat <- as.matrix(dist(x))
    kern <- match.arg(kern)
    W <- switch(
      kern,
      boxcar = dmat <= (band * 0.5),
      gaussian = dnorm(dmat, 0, sd = band * 0.3706506)
    )
    W <- sweep(W, 1, rowSums(W), '/')
```

```
    fit <- W %*% y
    yhat = W %*% y
    loocv_comp <- mean((y - yhat) ^ 2 / (1 - diag(W)) ^ 2)
    out <- list(fit = fit,
                edf = sum(diag(W)),
                loocv = loocv_comp)
    out
  }
```

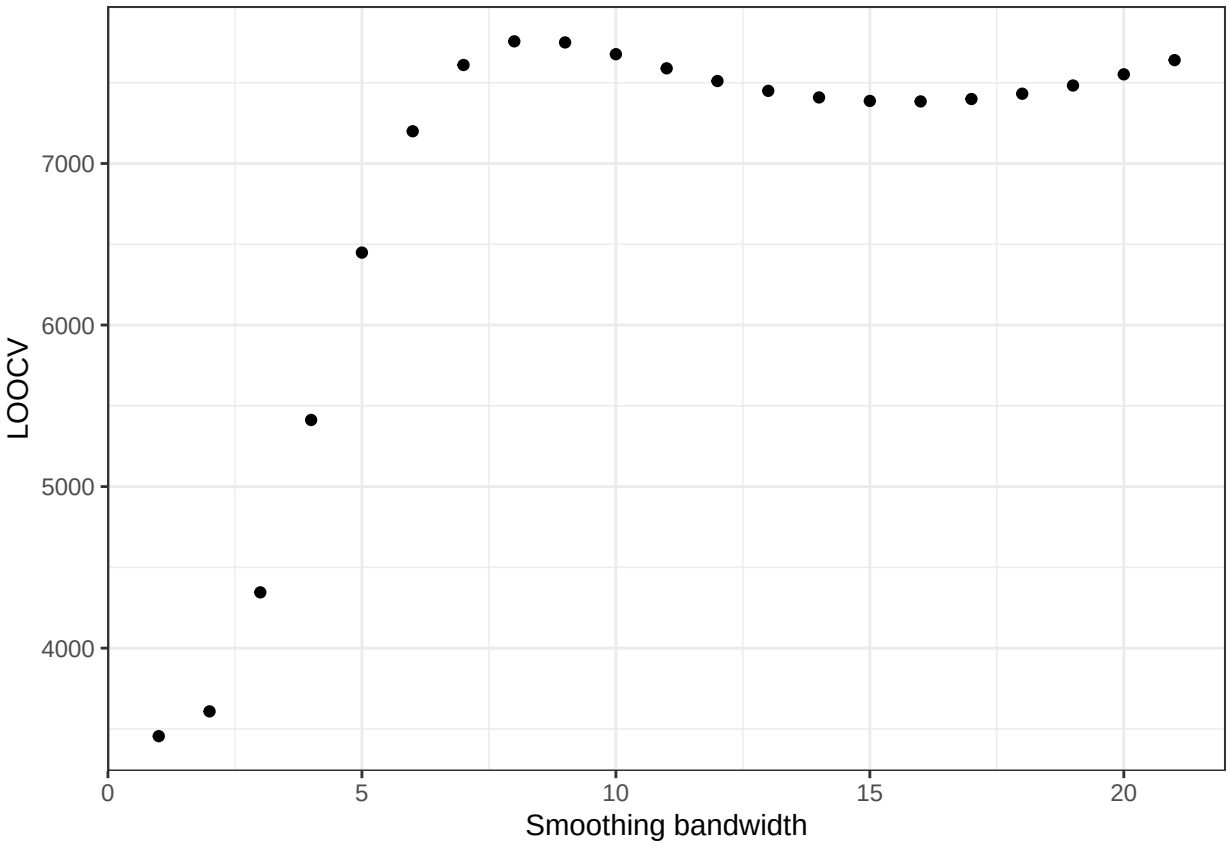
```
bws <- 1:21
band_loocv <- rep(NA, length(bws))

for (i in bws) {
  gaus_smooth <-
    kernel_smoother(bc_smoothed$date,
                    bc_smoothed$cases,
                    kern = "gaussian",
                    band = i)
  band_loocv[i] = gaus_smooth$loocv
}

band_loocv
```

```
##  [1] 3455.020 3608.477 4345.241 5412.440 6448.398 7199.485 7609.662 7756.013
##  [9] 7749.240 7676.489 7588.901 7510.397 7449.692 7408.727 7387.126 7384.095
## [17] 7399.078 7431.886 7482.662 7551.802 7639.878
```

```
library(FNN)
band_loocv.df <- data.frame(band_loocv)
ggplot(band_loocv.df, aes(bws, band_loocv))+geom_point()+
  ylab("LOOCV")+
  xlab("Smoothing bandwidth")
```



Which value is best? Will the resulting plot be smoother or wigglier than with `band = 7`*? Are there any reasons that this is best bandwidth by this metric? Should it be used?*

Above we can see a plot that shows how the smoothing bandwidth affects the quality of prediction. The LOOCV error seems to be increasing with higher width of bands. The smoothing bandwidth that minimizes this LOOCV error is `band = 1`. The resulting plot will be wiggler than with `band = 7`: the higher the band, the smoother the function since we consider more observations.
As the bandwidth increases, we consider more points, so the bias will go down and variance will go up. (This is because our sigma here increases as the width increases, hence, the variance also increases). Hence, the loocv-error will be becoming larger with increase in bandwidth. That is why if we need to select a model with the lowest loocv, the model with bandwidth 1 for this matrix will be the best. However, we should not use it for smoothing due to the fact that it gives a bad smoothing with a lot of fluctuations that distort the overall trend of the data. So, it is probably better to examine bandwidth larger than 7 to avoid wiggly behaviour. Hence, it looks like the best is `band = 16`.