Prediction

Jaime Davila/Matt Richey

2/16/2021

Classification and choice of K

On a first instance we will be exploring our Minneapolis police incidents dataset in a more systematic way and construct visualizations that explore the effect of the value of K in our KNN model.

Let's start by loading the dataset

```
mn.police.tbl <- read_csv("~/Mscs 341 S22/Class/Data/police_incidents.mn.csv")
```

And remember the steps that we took last time, namely:

1. Divide our dataset into training and testing datasets.
2. Construct a model based on the training dataset.
3. Evaluate the fit of the model by calculating the MSE on the testing dataset

These steps can be done as follows:

```
# Divide dataset into testing and training
train.mn.police.tbl <- mn.police.tbl %>%
  filter(year==2016)
test.mn.police.tbl <- mn.police.tbl %>%
  filter(year==2017)

# Build the KNN model
kNear=7
knn.model <- knnreg(tot~week, data=train.mn.police.tbl,k=kNear)

# Evaluate the MSE of the KNN model in the testing dataset
test.pred <- predict(knn.model, test.mn.police.tbl)
(mse.test <- mean ((test.mn.police.tbl$tot-test.pred)^2))
```

```
## [1] 155.8458
```

We are interested in calculating systematically the MSE as we iterate over the parameter k by doing the following steps:

1. Create a function `calc_MSE(kNear, train.tbl, test.tbl)` that trains a KNN model with parameter `kNear` on `train.tbl` and then applies the model on `test.tbl` and calculates the MSE. Test your function with k=7 and k=35 using our testing and training datasets.

```
calc_MSE <- function(kNear, train.tbl, test.tbl){

  knn.model <- knnreg(tot~week, data=train.tbl,k=kNear)
  test.predict <- predict(knn.model, test.tbl)
  test_MSE <- mean((test.predict - test.tbl$tot)^2)
```

```
  test_MSE

}

calc_MSE(7, train.mn.police.tbl, test.mn.police.tbl)
```

```
## [1] 155.8458
```

```
calc_MSE(35, train.mn.police.tbl, test.mn.police.tbl)
```

```
## [1] 149.0255
```

2. We would like to create a vector with the MSE values for our testing dataset. Notice it only makes sense to look at values of k in increments of 7 (why?). Use a for loop in R(Look at the syntax of for loops in https://rafalab.github.io/dsbook/programming-basics.html#for-loops) to create the mse for $k = 7, 14, 21, ..., 364$

It makes sense to increase k in values of 7 because there are 7 days in a week so we are clustering k number of weeks together

```
mse_test <- vector(length = 364/7)

for (i in seq(from = 7, to = 364, by = 7)){
  knn.model <- knnreg(tot~week, data=train.mn.police.tbl,k=i)
  test.predict <- predict(knn.model, test.mn.police.tbl)
  test_MSE <- mean((test.predict - test.mn.police.tbl$tot)^2)

  mse_test[i/7] = test_MSE
}

mse_test
```

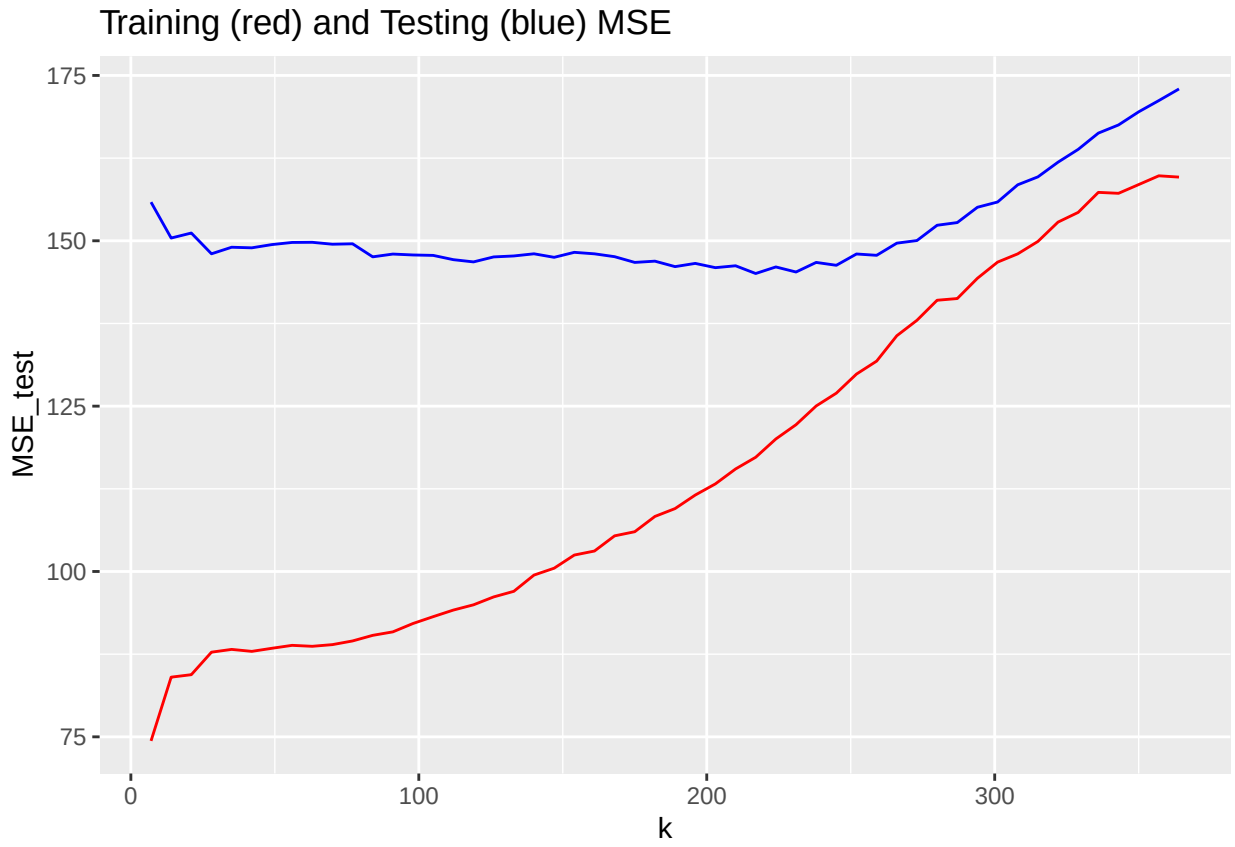
```
##  [1] 155.8458 150.4184 151.1748 148.0340 149.0255 148.9372 149.4296 149.7515
##  [9] 149.7698 149.4829 149.5480 147.5733 147.9949 147.8568 147.7932 147.1485
## [17] 146.8099 147.5607 147.7083 148.0340 147.5022 148.2566 148.0311 147.6000
## [25] 146.7298 146.9207 146.0966 146.5842 145.9356 146.2218 145.0544 146.0451
## [33] 145.2785 146.7315 146.2946 148.0077 147.8052 149.6433 150.0378 152.3477
## [41] 152.7568 155.0693 155.8649 158.4674 159.6662 161.8858 163.8238 166.2826
## [49] 167.5178 169.4999 171.2018 172.9603
```

3. Generate a graph depicting the MSE as a function of `k` for both testing and training datasets. What is the optimal value for `k` based on the testing dataset? Can you find this value in a systematic way? (*Hint*: Check the documentation for function `slice_min`). Compare this graph against figures 2.9 and 2.10 from your book. What would be the equivalent of the parameter `flexibility` in your KNN model?

```
mse_train <- vector(length = 364/7)

for (i in seq(from = 7, to = 364, by = 7)){
  knn.model <- knnreg(tot~week, data=train.mn.police.tbl,k=i)
  train.predict <- predict(knn.model, train.mn.police.tbl)
  train_MSE <- mean((train.predict - train.mn.police.tbl$tot)^2)

  mse_train[i/7] = train_MSE
}

mse_train
```

```
##  [1]  74.38533  84.02415  84.39819  87.80070  88.22261  87.92317  88.38951
##  [8]  88.83341  88.68916  88.94294  89.49829  90.34616  90.87219  92.13889
## [15]  93.17287  94.16757  94.96160  96.15722  97.00190  99.45887 100.49841
## [22] 102.48378 103.09478 105.38817 106.01265 108.32493 109.51318 111.55615
## [29] 113.24546 115.49922 117.27036 120.03796 122.19204 125.02854 126.96264
## [36] 129.84923 131.81150 135.64431 137.97729 141.00899 141.26744 144.33255
## [43] 146.77651 148.02694 149.90044 152.82722 154.29698 157.32222 157.17755
## [50] 158.50259 159.82773 159.62983
```

```
mse_tbl <- tibble(k = seq(from = 7, to = 364, by = 7), MSE_test = mse_test, MSE_train = mse_train)

mse_tbl%>%
  ggplot(aes(x = k))+
  #geom_point(aes(y = MSE_test), color = "blue")+
  geom_line(aes(y = MSE_test), color = "blue")+
    #geom_point(aes(y = MSE_train), color = "red")+
    geom_line(aes(y = MSE_train), color = "red")+
  labs(title = "Training (red) and Testing (blue) MSE")
```




```
slice_min(mse_tbl, MSE_test, n = 5)
```

```
## # A tibble: 5 x 3
##       k MSE_test MSE_train
##   <dbl>    <dbl>     <dbl>
## 1   217     145.      117.
## 2   231     145.      122.
## 3   203     146.      113.
```

```
## 4   224     146.      120.
## 5   189     146.      110.
```

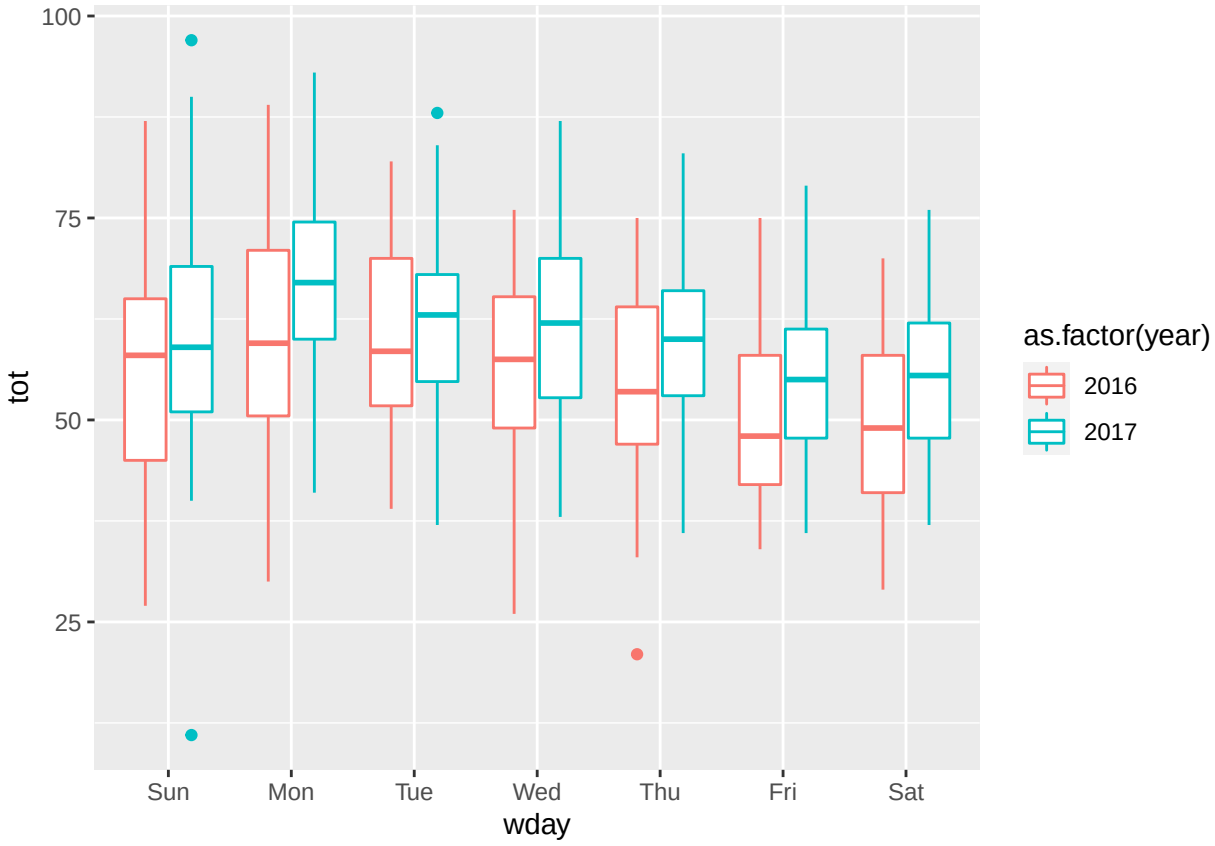
If we put all the k and MSE values into a tibble, we can use slice_min to quickly find that k=217 yields the lowest MSE. Relating to figures 2.9 and 2.10, our 'k' value would be similar to the flexibility in that graph. We see that the training MSE always increases as k increase, while the testing MSE generally decreases and then starts an increasing trend after k hits 217.

Improving your model

One way to improve the performance of a model is to use the information provided by other variables. In the following exercises we will explore this in more detail:

4. Does the distribution of number of incidents across the days of the week? Generate a boxplot to explore this question and check if this behavior is consistent across the years.

```
mn.police.tbl%>%
  mutate(wday = factor(wday, levels = c("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat")))%>%
ggplot()+
  geom_boxplot(aes(x = wday, y = tot, color = as.factor(year)))
```



We see that in general Monday and Tuesday have the highest number of incidents, with Sunday and Wednesday very close. We see Friday and Saturday as having the lowest number of incidents, regardless of year. In general, we see 2016 numbers as slightly below 2017 numbers but the general trends among day of the week seems pretty similar.

5. It seems police incidents are higher on Monday as opposed to other days. Subset your dataset to only

Mondays and construct a KNN model using `week` as input variable and train it using the data from 2016. Plot a graph with the MSE for all different choices of k and select a k that minimizes the MSE on the testing. Is the MSE smaller using this model than our original model?

```
#Subsetting the DATA
mon.police.tbl <- mn.police.tbl%>%
  filter(wday == "Mon")

train.mon.police.tbl <- mon.police.tbl%>%
  filter(year == "2016")

test.mon.police.tbl <- mon.police.tbl%>%
  filter(year == "2017")
```

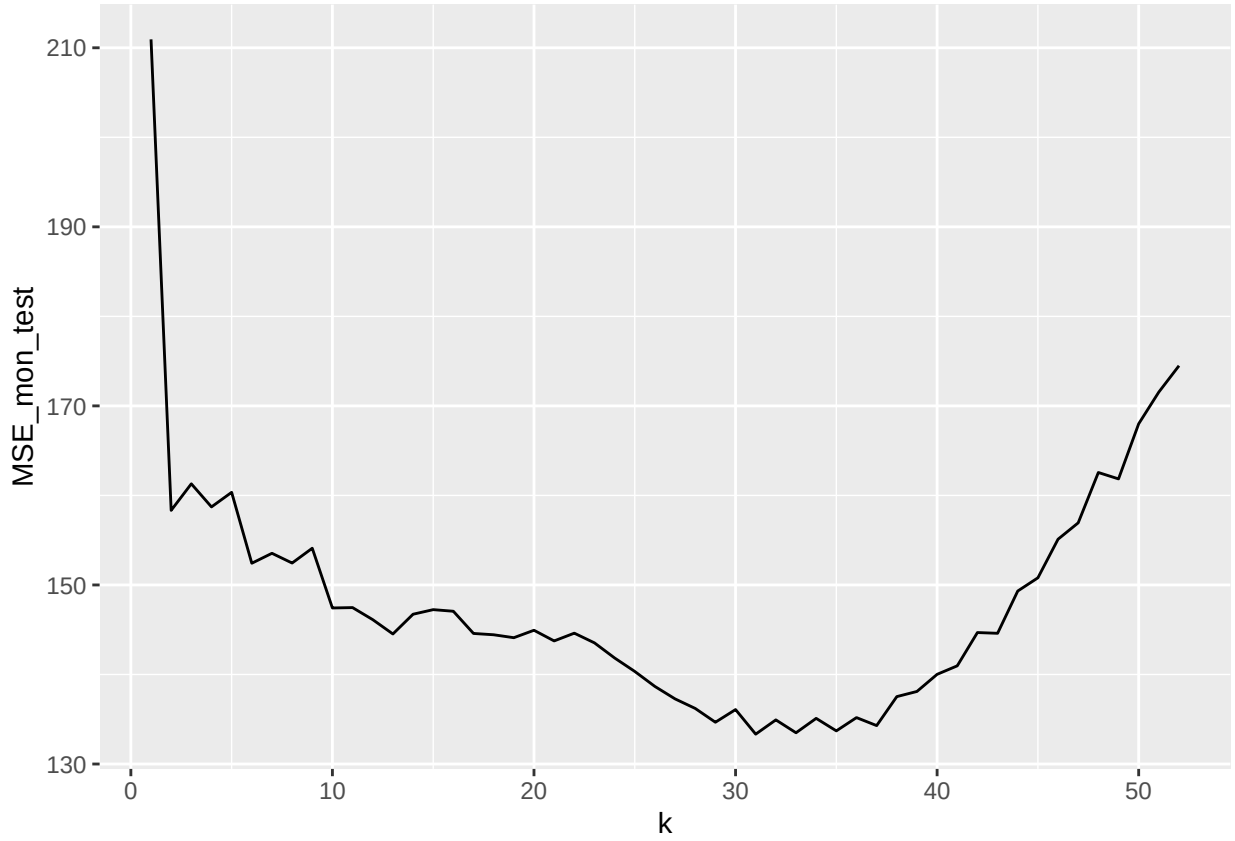
```
# kNear=7
# knn.model.mon <- knnreg(tot~week, data=train.mon.police.tbl,k=kNear)

#Iterating over k
mse_mon_test <- vector(length = 52)

for (i in 1:52){
  knn.model <- knnreg(tot~week, data=train.mon.police.tbl,k=i)
  test.predict <- predict(knn.model, test.mon.police.tbl)
  test_MSE <- mean((test.predict - test.mon.police.tbl$tot)^2)

  mse_mon_test[i] = test_MSE
}

mse_mon_test
```

```
##  [1] 210.9423 158.3317 161.2970 158.7174 160.3446 152.4250 153.5345 152.4472
##  [9] 154.1004 147.4307 147.4720 146.1247 144.5256 146.7305 147.2432 147.0620
## [17] 144.5818 144.4373 144.1089 144.9403 143.7472 144.6135 143.5313 141.8440
## [25] 140.3392 138.6632 137.2735 136.2068 134.6670 136.0805 133.3343 134.9272
## [33] 133.4887 135.0991 133.7003 135.1835 134.2919 137.5281 138.1022 140.0143
## [41] 140.9663 144.6841 144.6008 149.3212 150.7969 155.1085 156.9384 162.5611
## [49] 161.8453 167.9983 171.5482 174.4867
```

```
#Forming a tibble
mon_mse <- tibble(k = 1:52, MSE_mon_test= mse_mon_test)

#Finding k that minimizes MSE on testing
mon_mse%>%
  slice_min(MSE_mon_test, n = 1)
```

```
## # A tibble: 1 x 2
##       k MSE_mon_test
##   <int>        <dbl>
## 1    31         133.
```

```
#Plotting
mon_mse%>%
  ggplot(aes(x = k, y = MSE_mon_test))+
  geom_line()
```



We see a minimum MSE of 133, which is less than the 145 we found in question 3, so we have lowered our MSE.

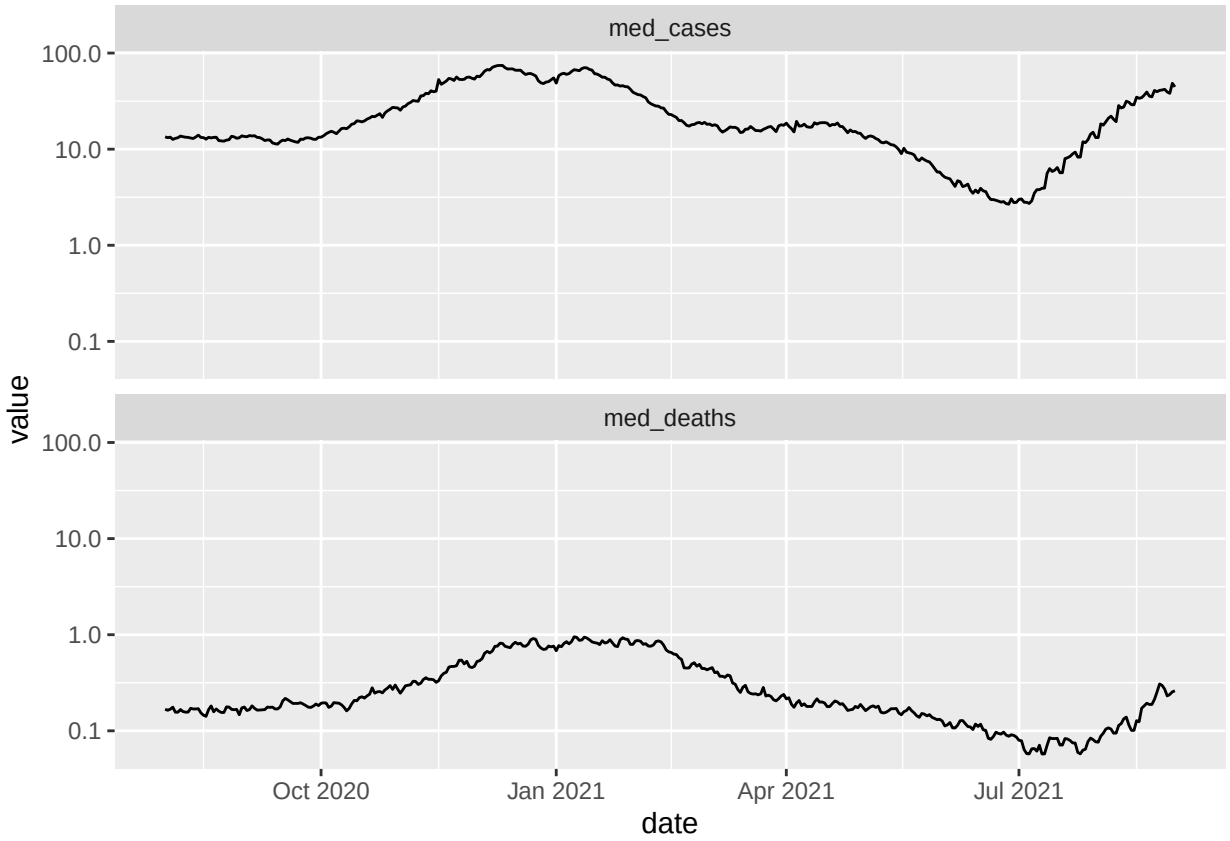
Studying the COVID pandemic

For the next set of exercises we will be using the US covid dataset procured from CovidActNow. We'll start by loading the dataset. Notice that I also loaded the library `lubridate` which allows for convenient use of dates.

```
library(lubridate)
covid.tbl <- read_csv("~/Mscs 341 S22/Class/Data/covid.csv")
```

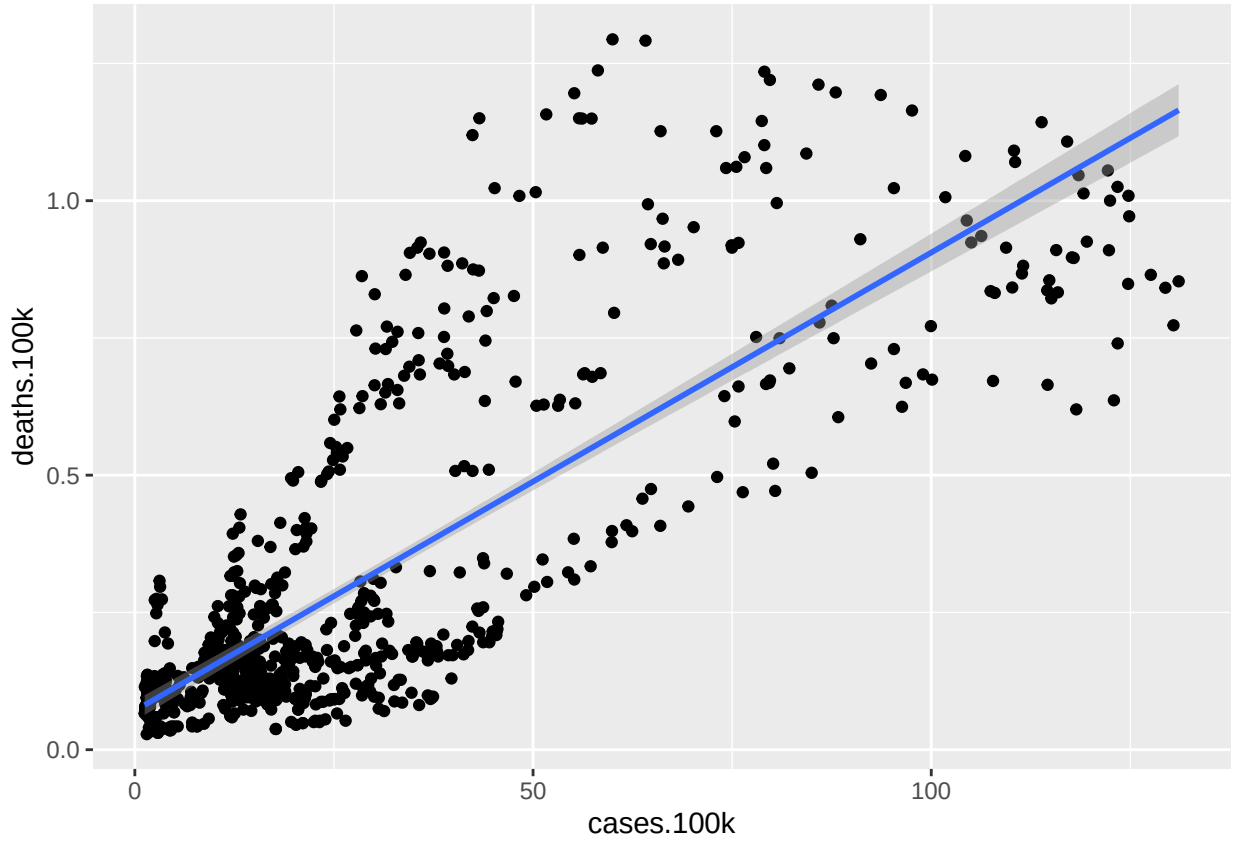
6. Subset your dataset from August 2020 to August 2021 and plot the median number of cases and the median number of deaths (per 100,000). *Hint*: You can filter dates using `filter` and you will need to create a reference date with `as.Date`

```
covid.tbl%>%
  filter(date >= as.Date("2020-08-01"), date <= as.Date("2021-08-31"))%>%
  group_by(date)%>%
  summarize(med_cases = median(cases.100k),
            med_deaths = median(deaths.100k))%>%
  pivot_longer(2:3, names_to = "Type", values_to = "value")%>%
  ggplot(aes(x = date, y = value))+
    geom_line()+
  facet_wrap(~Type, nrow = 2)+
  scale_y_log10()
```



7. We would like to create a model that would be able to predict the number of deaths based on the number of cases. To do that let's create training and testing datasets from neighboring states, let's say WI and MN. Plot the number of cases against the number of deaths for your training dataset and include a linear trend in your plot

```
new.covid.tbl<- covid.tbl%>%
  filter(date >= as.Date("2020-08-01"), date <= as.Date("2021-08-31"))

new.covid.tbl%>%
  filter(state == "MN"| state == "WI")%>%
  ggplot(aes(x = cases.100k, y = deaths.100k))+
    geom_point()+
    geom_smooth(method = "lm")
```



8. Create a linear model using `lm()` using the data from WI (training) and evaluate how well it does in MN (testing).

```
wi.covid <- new.covid.tbl%>%
  filter(state == "WI")

mn.covid <- new.covid.tbl%>%
  filter(state == "MN")

wi_model <- lm(deaths.100k ~ cases.100k, data = wi.covid)



test.predict <- predict(wi_model, mn.covid)
test_MSE <- mean((test.predict - mn.covid$deaths.100k)^2)

test_MSE
```

```
## [1] 0.03737339
```

9. To improve our model, let's make use of the fact that the number of covid cases is a good predictor of the number of deaths a couple of weeks afterwards. Create a function `calc_MSE_lag(time.lag, train.tbl, test.tbl)` that calculates the MSE on the testing dataset by using a linear model where the deaths lag the number of cases by `time.lag` *Hint* Make use of the functions `lead/lag` from the tidyverse.

```
lag_test <- wi.covid%>%
  mutate(lead_date = lead(date, n = 7),
```

```
         lead_deaths = lead(deaths.100k, n = 7))


test_model <- lm(lead_deaths ~ cases.100k, data = lag_test)



calc_MSE_lag <- function(time.lag, train.tbl, test.tbl){
  lead_train_tbl <-train.tbl%>%
    mutate(lead_date = lead(date, n = time.lag),
           lead_deaths = lead(deaths.100k, n = time.lag))

  lead_test_tbl <- test.tbl%>%
    mutate(lead_date = lead(date, n = time.lag),
           lead_deaths = lead(deaths.100k, n = time.lag))

  lead_model <- lm(lead_deaths ~ cases.100k, data = lead_train_tbl)

  test.predict <- predict(lead_model, lead_test_tbl)
  test_MSE <- mean((test.predict - lead_test_tbl$lead_deaths)^2, na.rm = TRUE)

  test_MSE

}

calc_MSE_lag(7, wi.covid, mn.covid)
```

```
## [1] 0.02200948
```

```
calc_MSE_lag(14, wi.covid, mn.covid)
```

```
## [1] 0.01464344
```

```
calc_MSE_lag(21, wi.covid, mn.covid)
```

```
## [1] 0.01617032
```

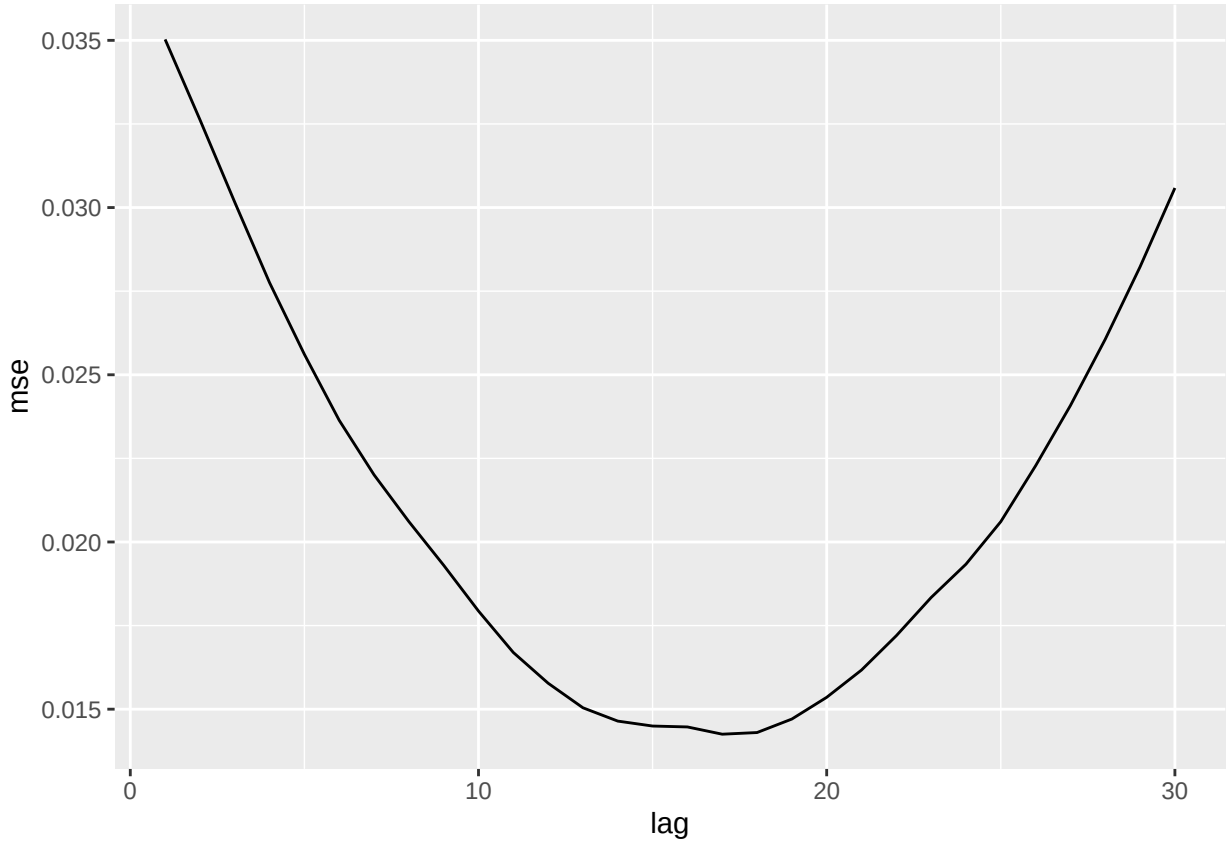
10. Plot lag versus MSE on the testing dataset and find the optimal parameter of lag. How do you interpret this optimal lag? Using this value of lag, plot the lagged number of cases versus the deaths and the linear trend line for the testing dataset.

```
#Finding MSE for all lag values from 1-30
lag_MSE <- vector(length = 30)

for(i in 1:30){
  lag_MSE[i] = calc_MSE_lag(i, wi.covid, mn.covid)
}


#Converting into a tibble and finding min MSE
lag_mse_tbl <- tibble(lag = 1:30, mse = lag_MSE)
lag_mse_tbl%>%
  slice_min(mse, n = 1)
```

```
## # A tibble: 1 x 2
##     lag    mse
```

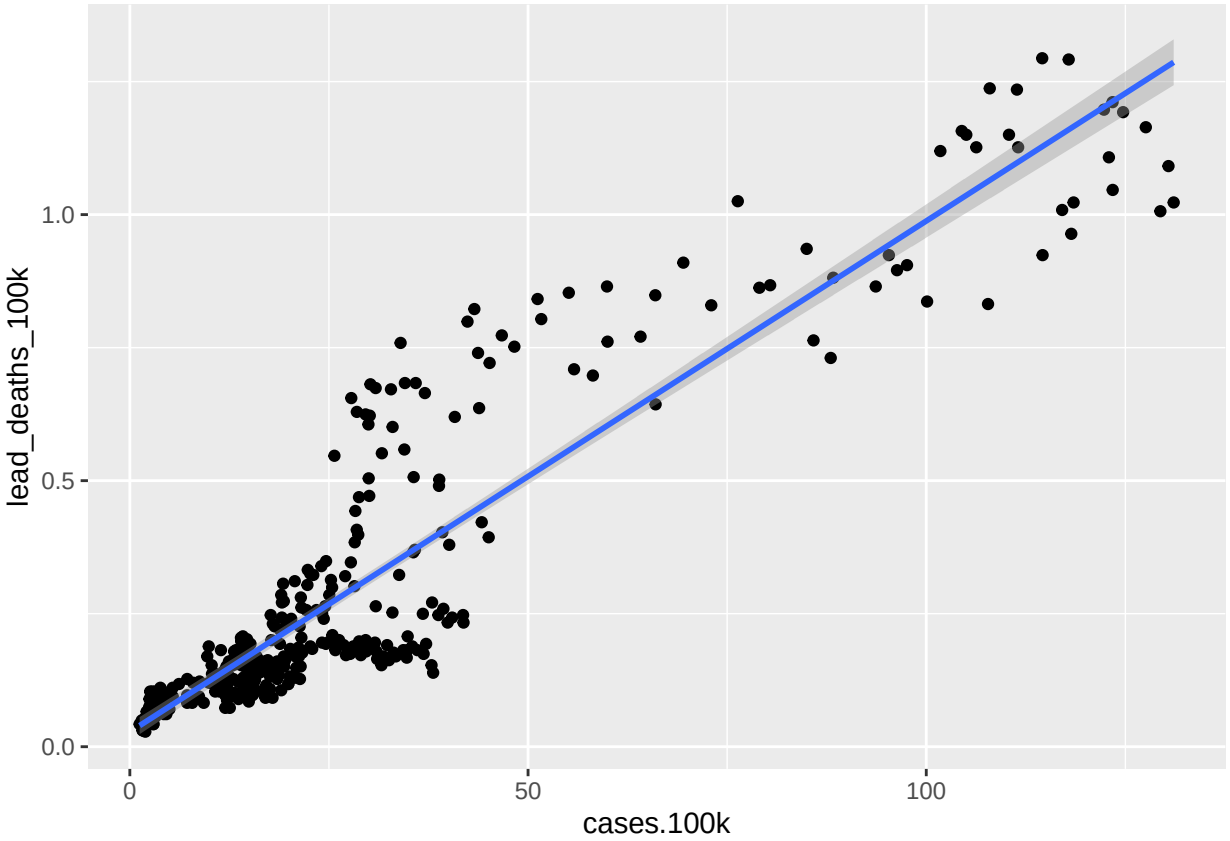
```
##   <int>  <dbl>
## 1    17 0.0143
```

```
lag_mse_tbl%>%
  ggplot(aes(x = lag, y = mse))+
  geom_line()
```



We see that 17 represents the optimal lag, which means that we can best predict deaths using cases if we align deaths numbers with the case numbers that came 17 days beforehand. This is plotted below.

```
mn.covid%>%
  mutate(lead_date = lead(date, n = 17),
           lead_deaths_100k = lead(deaths.100k, n = 17))%>%
  ggplot(aes(x = cases.100k, y = lead_deaths_100k))+
    geom_point()+
    geom_smooth(method = "lm")
```


lead_deaths_100k
1.0
0.5
0.0
0
50
100
cases.100k