Prediction

Ivana K.

2/16/2021

Notes:

"Prediction" means the response variable is continuous

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

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

```
## # A tibble: 6 x 4
##   wday   week  year   tot
##   <chr> <dbl> <dbl> <dbl>
## 1 Sun       1  2016    50
## 2 Sun       1  2017    53
## 3 Sun       2  2016    43
## 4 Sun       2  2017    51
## 5 Sun       3  2016    42
## 6 Sun       3  2017    54
```

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:

```r
# 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
```

```r
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.

```r
calc_MES <- function(kNear, train.tbl, test.tbl) {
  #train KNN model on train.tbl
  knn.model <- knnreg(tot~week, data=train.tbl,k=kNear)
  #apply to test.tbl
  test.pred <- predict(knn.model, test.tbl)
  #calculate MSE
  (mse.test <- mean ((test.tbl$tot-test.pred)^2))
}

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

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

```r
calc_MES(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 $k = 7, 14, 21, ..., 364$

```r
mseVector <- function(startVal, endVal) {
  vector = c()
  for(i in seq(from=startVal, to=endVal, by=7)){
  vector[i/7] = calc_MES(i,train.mn.police.tbl, test.mn.police.tbl)
  }
  vector
}
testDataSet <- mseVector(7,364)

mseVector <- function(startVal, endVal) {
  vector = c()
  for(i in seq(from=startVal, to=endVal, by=7)){
  vector[i/7] = calc_MES(i,train.mn.police.tbl, train.mn.police.tbl)
  }
  vector
}
trainDataSet <- mseVector(7,364)
```

"Notice it only makes sense to look at values of k in increments of 7 (why?)" not sure.

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?

generate graph:

```
testDataSet
```

```
##  [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
```

```
trainDataSet
```

```
##  [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
```

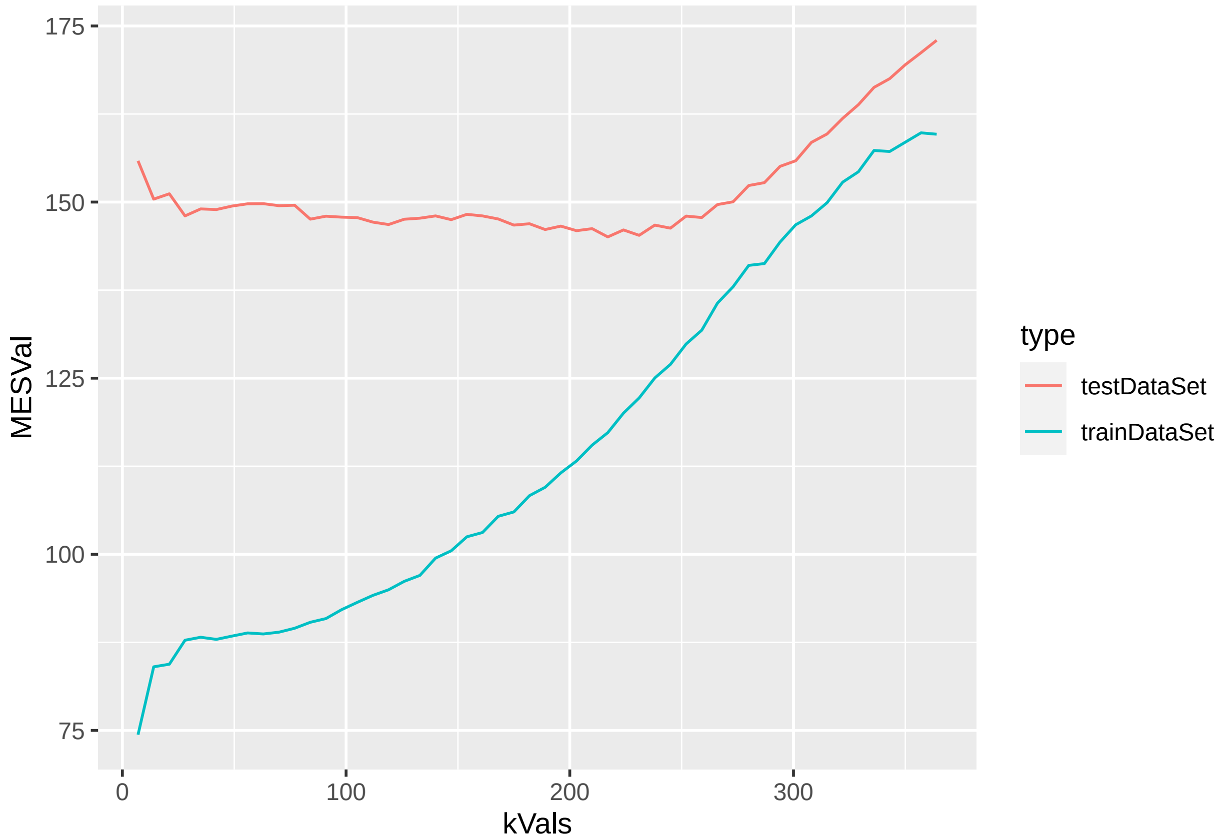
```
kVals = c()
for(i in seq(from=7, to=364, by=7)){
  kVals[i/7] = i
}
kVals
```

```
##  [1]   7  14  21  28  35  42  49  56  63  70  77  84  91  98 105 112 119 126 133
## [20] 140 147 154 161 168 175 182 189 196 203 210 217 224 231 238 245 252 259 266
## [39] 273 280 287 294 301 308 315 322 329 336 343 350 357 364
```

plot it and get into tibble:

```
finally <- as_tibble(data.frame(kVals, testDataSet, trainDataSet))
finally2 <- finally %>%
  mutate(diffVal = testDataSet - trainDataSet) %>%
  pivot_longer(cols = testDataSet:trainDataSet, names_to = "type", values_to = "MESVal")

finally2 %>%
  mutate(type = as.factor(type)) %>%
  ggplot(mapping = aes(kVals, MESVal)) +
  geom_line(mapping = aes(kVals, MESVal, color = type))
```



find value of k in systematic way. use slice_min

```
finally2
```

```
## # A tibble: 104 x 4
##    kVals diffVal type         MESVal
##    <dbl>   <dbl> <chr>         <dbl>
##  1     7    81.5 testDataSet   156.
##  2     7    81.5 trainDataSet   74.4
##  3    14    66.4 testDataSet   150.
##  4    14    66.4 trainDataSet   84.0
##  5    21    66.8 testDataSet   151.
##  6    21    66.8 trainDataSet   84.4
##  7    28    60.2 testDataSet   148.
##  8    28    60.2 trainDataSet   87.8
##  9    35    60.8 testDataSet   149.
## 10    35    60.8 trainDataSet   88.2
## # ... with 94 more rows
```

```
finally2 %>%
  filter(type == "testDataSet") %>%
  slice_min(MESVal, with_ties = FALSE)
```

```
## # A tibble: 1 x 4
##   kVals diffVal type        MESVal
##   <dbl>   <dbl> <chr>        <dbl>
## 1   217    27.8 testDataSet   145.
```

read plot on figure 2.9 in book. and the thing with flexibility It should be 217. why?

Improving your model

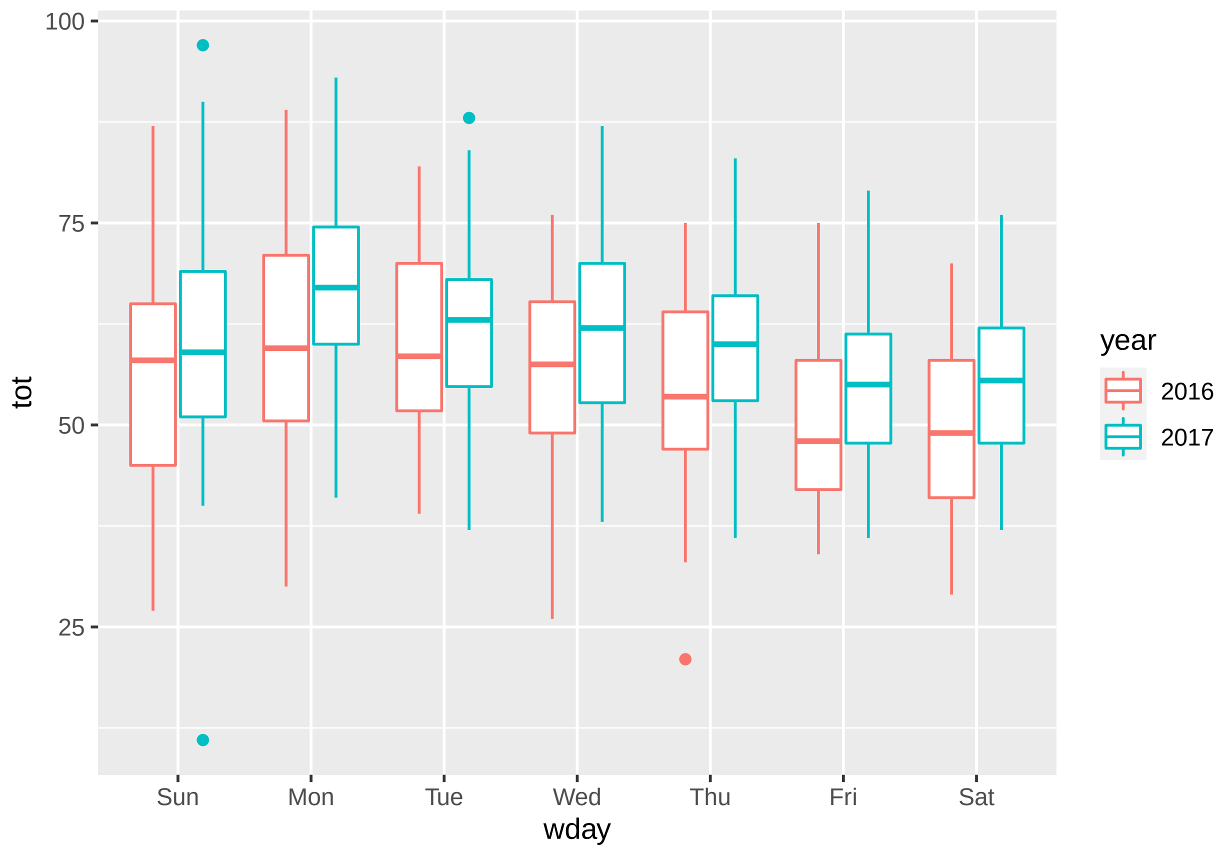
One way to improve the performance of a model is to use the information provided by other variables. In the following exercises we will explore this in more detail:

(day of the week)

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

```
days = unique(mn.police.tbl$wday)

mn.police.tbl %>%
  mutate(year = as.factor(year),
         wday = factor(wday, levels = days)) %>%#factor puts it in certain order
    ggplot(aes(x = wday, y = tot,
               color = year)) +
      geom_boxplot()
```



5. It seems police incidents are higher on Monday as opposed to other days. Subset you 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?

```r
# Divide dataset into testing and training
days = unique(mn.police.tbl$wday)

trainDayTbl <- mn.police.tbl %>%
  mutate(year = as.factor(year),
         wday = factor(wday, levels = days)) %>%
  filter(year == 2016) %>%#   train KNN using data from 2016
  filter(wday == "Mon")#  subset data to only include Mondays

trainDayTbl
```

```
## # A tibble: 52 x 4
##    wday   week year    tot
##    <fct> <dbl> <fct> <dbl>
##  1 Mon       1 2016     53
##  2 Mon       2 2016     42
##  3 Mon       3 2016     30
##  4 Mon       4 2016     70
##  5 Mon       5 2016     33
##  6 Mon       6 2016     49
##  7 Mon       7 2016     40
##  8 Mon       8 2016     51
##  9 Mon       9 2016     34
## 10 Mon      10 2016     54
## # ... with 42 more rows
```

```r
testDayTbl <- mn.police.tbl %>%
  mutate(year = as.factor(year),
         wday = factor(wday, levels = days)) %>%
  filter(year == 2017) %>%
  filter(wday == "Mon")

testDayTbl
```

```
## # A tibble: 52 x 4
##    wday   week year    tot
##    <fct> <dbl> <fct> <dbl>
##  1 Mon       1 2017     56
##  2 Mon       2 2017     53
##  3 Mon       3 2017     50
##  4 Mon       4 2017     56
##  5 Mon       5 2017     71
##  6 Mon       6 2017     52
##  7 Mon       7 2017     63
##  8 Mon       8 2017     48
##  9 Mon       9 2017     61
## 10 Mon      10 2017     69
## # ... with 42 more rows
```

```r
# Build the KNN model
kNear=14
knn.model2 <- knnreg(tot~week, data=trainDayTbl,kNear)#week as input variable

# Evaluate the MSE of the KNN model in the testing dataset
test.pred2 <- predict(knn.model2, testDayTbl)
```

```
(mse.test <- mean((testDayTbl$tot-test.pred2)^2))
```

```
## [1] 823.7115
```

put stuff into function so a vector of MES values can be calculated:

```
calc_MES2 <- function(kValue, traintable, testtable) {
  #train KNN model on train.tbl
  knn.model2 <- knnreg(tot~week, data=traintable,k=kValue)
  #apply to test.tbl
  test.pred2 <- predict(knn.model2, testtable)
  #calculate MSE
  (mse.test <- mean ((testtable$tot-test.pred2)^2))
}

#calc_MES2(7,trainDayTbl, testDayTbl)#138.6318
#calc_MES2(35,trainDayTbl, testDayTbl)
```

```
#test
mseVector <- function(startVal, endVal) {
  vector = c()
  for(i in seq(from=startVal, to=endVal)){#, by=7
  vector[i] = calc_MES2(i,trainDayTbl, testDayTbl)#[i/7]
  }
  vector
}
testDataSet2 <- mseVector(7,52)
#testDataSet2

#train
mseVector <- function(startVal, endVal) {
  vector = c()
  for(i in seq(from=startVal, to=endVal)){
  vector[i] = calc_MES2(i,trainDayTbl, trainDayTbl)
  }
  vector
}
trainDataSet2 <- mseVector(7,52)
#trainDataSet2
```

generate graph:

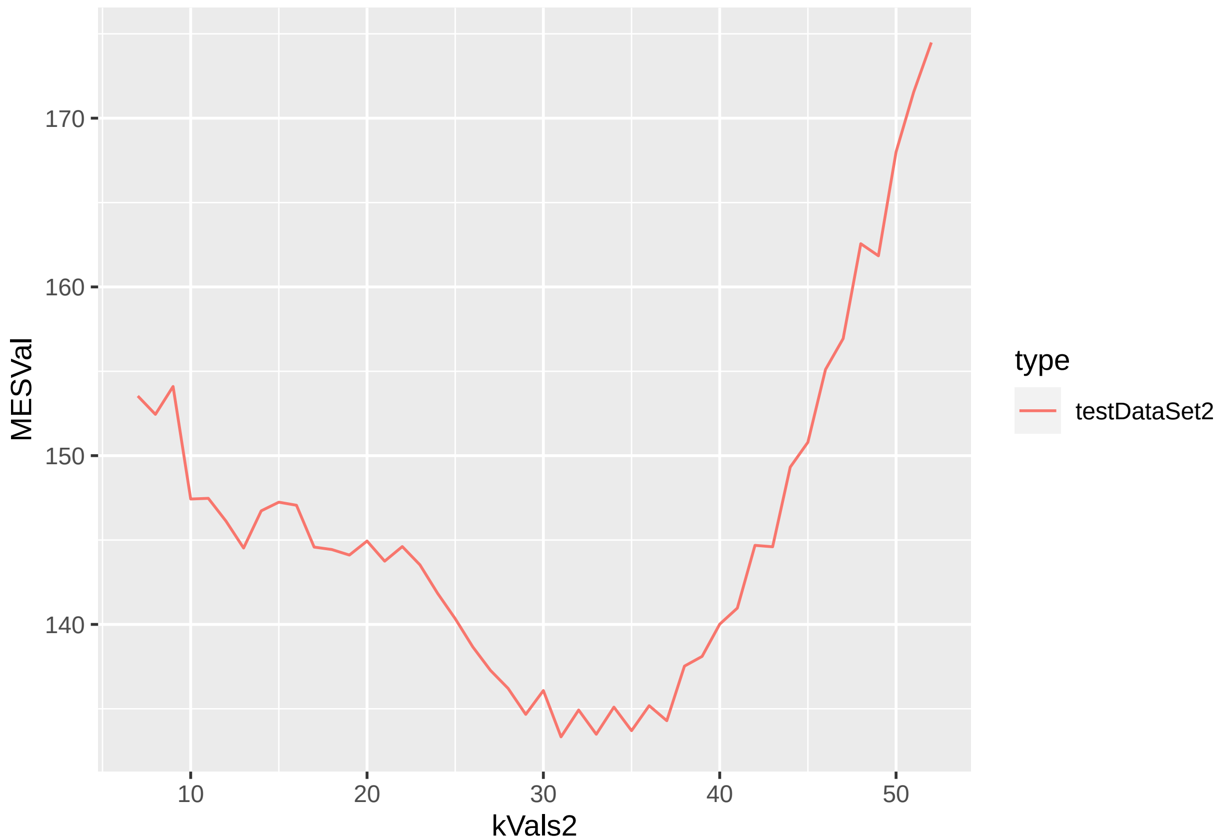
```
kVals2 = c()
for(i in seq(from=7, to=52)){#, by=7
  kVals2[i] = i#[i/7]
}
#kVals2
```

get into tibble for plotting:

```
tibble1 <- as_tibble(data.frame(kVals2, testDataSet2, trainDataSet2))
tibble2 <- tibble1 %>%
  mutate(diffVal = testDataSet2 - trainDataSet2) %>%
  pivot_longer(cols = testDataSet2:trainDataSet2, names_to = "type", values_to = "MESVal")
#tibble2

tibble2 %>%
```

```r
  mutate(type = as.factor(type)) %>%
  filter(type == "testDataSet2") %>%
  ggplot(mapping = aes(kVals2, MESVal)) +
  geom_line(mapping = aes(kVals2, MESVal, color = type))
```



find value of k in systematic way. use slice_min

```r
tibble2 %>%
  filter(type == "testDataSet2") %>%
  slice_min(MESVal, with_ties = FALSE)
```

```
## # A tibble: 1 x 4
##   kVals2 diffVal type          MESVal
##    <int>   <dbl> <chr>          <dbl>
## 1     31   -3.93 testDataSet2   133.
```

original: kVals diffVal type MESVal 1 217 27.8 testDataSet 145.

MSE value is smaller in this model.

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.

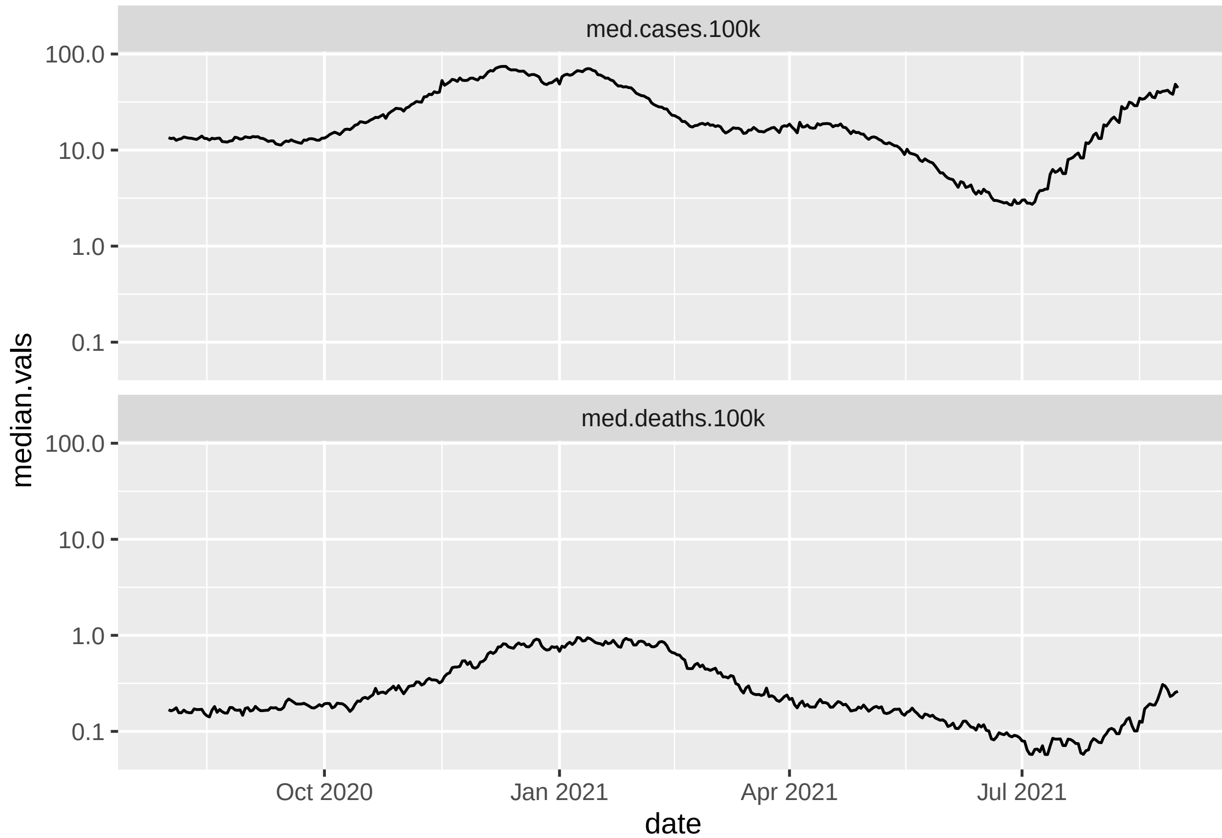
```r
library(lubridate)
covid.tbl <- read_csv("~/Mscs 341 S22/Class/Data/covid.csv")
```

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`

```r
covid.tbl
```

```
## # A tibble: 26,418 x 5
##    state region date       cases.100k deaths.100k
##    <chr> <chr>  <date>          <dbl>       <dbl>
##  1 AL    South  2020-04-01       2.28       0.293
##  2 AL    South  2020-04-02       2.83       0.188
##  3 AL    South  2020-04-03       3.74       0.167
##  4 AL    South  2020-04-04       3.31       0.157
##  5 AL    South  2020-04-05       3.52       0.130
##  6 AL    South  2020-04-06       3.51       0.136
##  7 AL    South  2020-04-07       3.58       0.149
##  8 AL    South  2020-04-08       3.92       0.139
##  9 AL    South  2020-04-09       4.52       0.131
## 10 AL    South  2020-04-10       4.55       0.126
## # ... with 26,408 more rows
```

```r
covid.tbl %>%
  filter(date >= as.Date("2020-08-01")) %>%
  filter(date <= as.Date("2021-08-31")) %>%
  group_by(date) %>%
  summarize(med.cases.100k = median(cases.100k),
            med.deaths.100k = median(deaths.100k)) %>%
  pivot_longer(med.cases.100k:med.deaths.100k, names_to = "type", values_to = "median.vals") %>%
  mutate(type=as.factor(type)) %>%
  ggplot() +
    geom_line(mapping = aes(date, median.vals)) +
    facet_wrap(~ type, ncol = 1) +
    scale_y_continuous(trans='log10')
```

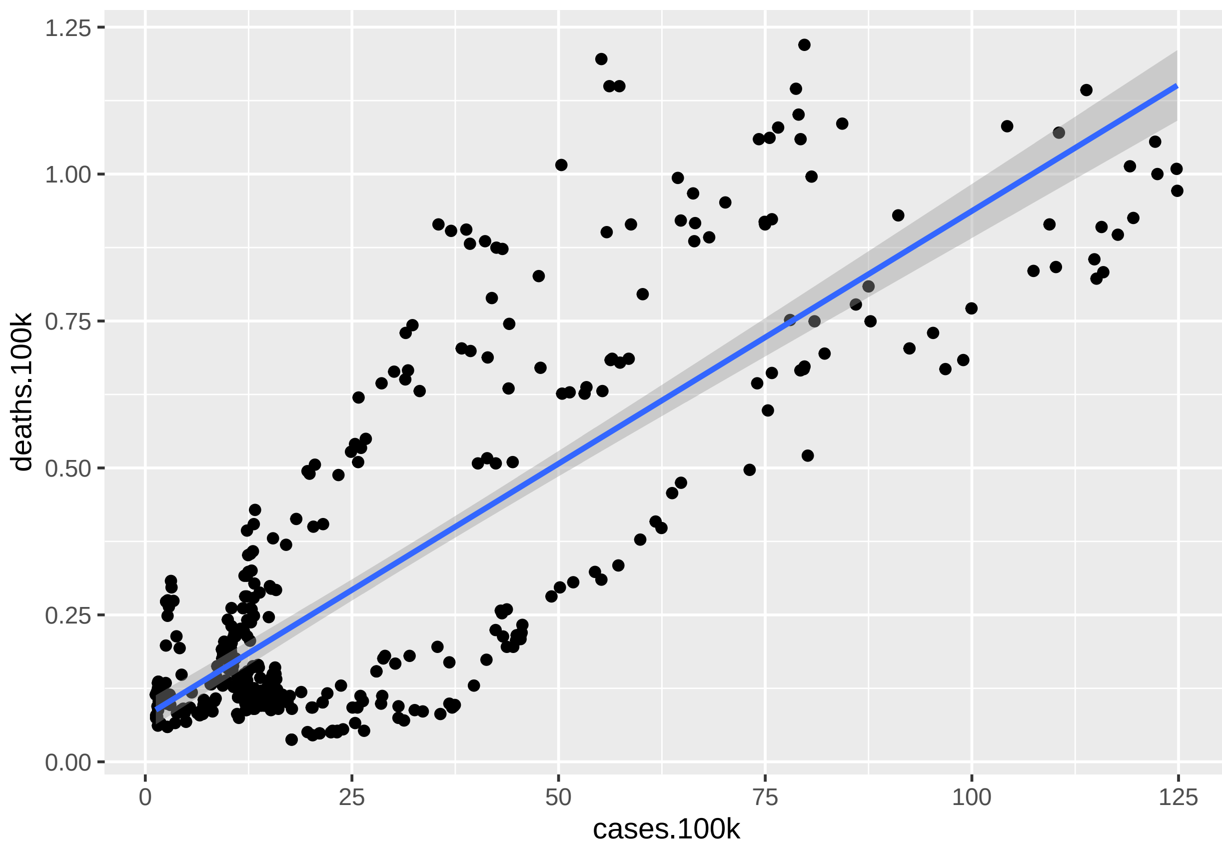


7. We would like to create a model that would be able to predict the number of deaths based on the number of cases. To do that let's create training and testing datasets from neighboring states, let's say WI and MN. Plot the number of cases against the number of deaths for your training dataset and include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

```
covid.train.tbl <- covid.tbl %>%#   train tbl: WI
  filter(state == "WI") %>%
  filter(date >= as.Date("2020-08-01")) %>%
  filter(date <= as.Date("2021-08-31"))

covid.test.tbl <- covid.tbl %>%
  filter(state == "MN") %>%
  filter(date >= as.Date("2020-08-01")) %>%
  filter(date <= as.Date("2021-08-31"))

covid.train.tbl %>%
  ggplot() +
    geom_point(mapping = aes(cases.100k, deaths.100k)) +
    geom_smooth(mapping = aes(cases.100k, deaths.100k), method='lm', se = TRUE)
```



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

```
linear.model <- lm(deaths.100k~cases.100k, data=covid.train.tbl)
linear.model
```

```
##
## Call:
## lm(formula = deaths.100k ~ cases.100k, data = covid.train.tbl)
##
## Coefficients:
## (Intercept)   cases.100k
##    0.077465     0.008597
```

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

test.pred.covid <- predict(linear.model, covid.test.tbl)
test.pred.covid
```

```
##          1          2          3          4          5          6          7
## 0.19295536 0.19206386 0.18701877 0.18604622 0.18888282 0.19283379 0.18878151
##          8          9         10         11         12         13         14
## 0.19171941 0.19317823 0.19032138 0.18444557 0.18175080 0.18320962 0.18055538
##         15         16         17         18         19         20         21
## 0.18349328 0.17990701 0.17518610 0.16965474 0.17439591 0.17887368 0.18160897
```

```
##          22         23         24         25         26         27         28 
## 0.18169002 0.18223707 0.18173054 0.17859002 0.18237890 0.19447496 0.19771679 
##          29         30         31         32         33         34         35 
## 0.20160698 0.20561874 0.20476776 0.20024947 0.20681416 0.21706643 0.21076514 
##          36         37         38         39         40         41         42 
## 0.21206187 0.20578083 0.19982398 0.19390764 0.18922726 0.18209524 0.17052598 
##          43         44         45         46         47         48         49 
## 0.17111356 0.16724363 0.16584560 0.16106390 0.16266456 0.17581421 0.19007824 
##          50         51         52         53         54         55         56 
## 0.19913509 0.20772592 0.21181873 0.20861743 0.21420957 0.22332721 0.22875726 
##          57         58         59         60         61         62         63 
## 0.23655791 0.23981999 0.23187752 0.22952720 0.23374157 0.24160300 0.24674939 
##          64         65         66         67         68         69         70 
## 0.25169318 0.24296051 0.24089385 0.24164352 0.24371018 0.25552258 0.26208728 
##          71         72         73         74         75         76         77 
## 0.26917877 0.26956374 0.27223824 0.27552059 0.28419247 0.28929835 0.30994472 
##          78         79         80         81         82         83         84 
## 0.31610419 0.32027804 0.32406692 0.32246627 0.32094667 0.32483686 0.33594011 
##          85         86         87         88         89         90         91 
## 0.33531200 0.33502834 0.33194861 0.34282899 0.35936229 0.39597466 0.42827134 
##          92         93         94         95         96         97         98 
## 0.45453012 0.45333470 0.47902616 0.51764441 0.55133913 0.59255083 0.64472395 
##          99        100        101        102        103        104        105 
## 0.67495397 0.73373231 0.76870350 0.80811194 0.83678183 0.90563009 0.93808886 
##         106        107        108        109        110        111        112 
## 1.00387765 1.06275730 1.09373699 1.13436111 1.13843366 1.19865056 1.18991789 
##         113        114        115        116        117        118        119 
## 1.20426297 1.17419504 1.14967874 1.12899184 1.13829183 1.03508022 0.99121427 
##         120        121        122        123        124        125        126 
## 1.03635669 1.09094092 1.06235207 1.00574170 0.98039468 0.97551168 1.08384943 
##         127        128        129        130        131        132        133 
## 1.09620889 1.02648938 0.95241368 0.89683664 0.91642942 0.88267392 0.83406680 
##         134        135        136        137        138        139        140 
## 0.81548709 0.75707345 0.70514347 0.64502787 0.62875796 0.59315867 0.57733451 
##         141        142        143        144        145        146        147 
## 0.55699206 0.52161564 0.49258105 0.46587651 0.44936346 0.44198831 0.38608709 
##         148        149        150        151        152        153        154 
## 0.37441652 0.36975640 0.33719632 0.31652968 0.32254732 0.33646690 0.29829441 
##         155        156        157        158        159        160        161 
## 0.34955577 0.36110478 0.37382894 0.38387860 0.41149490 0.41121124 0.41483803 
##         162        163        164        165        166        167        168 
## 0.46486344 0.45734646 0.42225370 0.38519559 0.38345311 0.36817601 0.36096295 
##         169        170        171        172        173        174        175 
## 0.34295056 0.31993360 0.29582252 0.29458658 0.29274279 0.28834606 0.28682646 
##         176        177        178        179        180        181        182 
## 0.28506372 0.27890425 0.26798335 0.26267486 0.26081081 0.26267486 0.25951408 
##         183        184        185        186        187        188        189 
## 0.25072063 0.23953633 0.23021609 0.22636642 0.22488734 0.23669974 0.23126968 
##         190        191        192        193        194        195        196 
## 0.22881805 0.22535335 0.21676251 0.21374356 0.21469585 0.22004486 0.21271023 
##         197        198        199        200        201        202        203 
## 0.21052200 0.20612527 0.20006711 0.19779783 0.20156646 0.20687495 0.20884030 
##         204        205        206        207        208        209        210 
## 0.20523377 0.20399783 0.19952005 0.19745339 0.20357234 0.20819194 0.21293311 
```

```
##        211        212        213        214        215        216        217 
## 0.20922527 0.20788802 0.20296449 0.20033051 0.20598344 0.20975207 0.20590240 
##        218        219        220        221        222        223        224 
## 0.20182985 0.20351155 0.19678476 0.21682330 0.22654877 0.23291085 0.23619320 
##        225        226        227        228        229        230        231 
## 0.24411541 0.24514874 0.24375071 0.24853240 0.23659843 0.24857292 0.25550232 
##        232        233        234        235        236        237        238 
## 0.26113499 0.25635330 0.25949382 0.26008140 0.27227877 0.28911600 0.29322906 
##        239        240        241        242        243        244        245 
## 0.29932775 0.29618723 0.30832381 0.31077544 0.32684274 0.34361918 0.35717406 
##        246        247        248        249        250        251        252 
## 0.36456948 0.32939567 0.33259698 0.36221915 0.37686815 0.39461714 0.40490994 
##        253        254        255        256        257        258        259 
## 0.40314720 0.39731191 0.43769289 0.43718636 0.41054261 0.42539422 0.42059227 
##        260        261        262        263        264        265        266 
## 0.41599293 0.40375504 0.39378643 0.37741521 0.38231847 0.38784983 0.37640214 
##        267        268        269        270        271        272        273 
## 0.37267404 0.35518844 0.34205905 0.33788520 0.34440938 0.35105512 0.34911002 
##        274        275        276        277        278        279        280 
## 0.33995187 0.33200940 0.32287150 0.31877870 0.32208131 0.32499895 0.31531400 
##        281        282        283        284        285        286        287 
## 0.30331924 0.28490162 0.27434543 0.26360689 0.26186440 0.25716376 0.25015331 
##        288        289        290        291        292        293        294 
## 0.23844222 0.22877753 0.22444159 0.21078540 0.21238605 0.21151481 0.20482854 
##        295        296        297        298        299        300        301 
## 0.19046321 0.18169002 0.17486192 0.16785148 0.16614952 0.16300900 0.15273647 
##        302        303        304        305        306        307        308 
## 0.14582733 0.13940446 0.13050970 0.12372213 0.12139207 0.11742083 0.11243653 
##        309        310        311        312        313        314        315 
## 0.11111954 0.10949862 0.10777640 0.11030908 0.11043065 0.11109927 0.10957967 
##        316        317        318        319        320        321        322 
## 0.10745222 0.10445353 0.10157641 0.09946923 0.09867903 0.09853720 0.09744309 
##        323        324        325        326        327        328        329 
## 0.09651106 0.09559930 0.09464701 0.09389734 0.09351237 0.09422152 0.09341107 
##        330        331        332        333        334        335        336 
## 0.09264113 0.09227643 0.09183068 0.09142545 0.09120257 0.09144571 0.09104048 
##        337        338        339        340        341        342        343 
## 0.09142545 0.09002741 0.08806205 0.08931826 0.09239800 0.09353263 0.09466727 
##        344        345        346        347        348        349        350 
## 0.09262087 0.09037185 0.09416074 0.10198164 0.10311628 0.10267053 0.10514242 
##        351        352        353        354        355        356        357 
## 0.10232608 0.10232608 0.10759405 0.11638750 0.11409796 0.11691430 0.12019665 
##        358        359        360        361        362        363        364 
## 0.11521234 0.11521234 0.12378291 0.13936394 0.13932341 0.14479399 0.15212862 
##        365        366        367        368        369        370        371 
## 0.14461164 0.14461164 0.15725476 0.18241943 0.18047433 0.18537759 0.19471810 
##        372        373        374        375        376        377        378 
## 0.18053512 0.18053512 0.20318737 0.22468472 0.22387427 0.23153308 0.24091411 
##        379        380        381        382        383        384        385 
## 0.22057166 0.22057166 0.25013305 0.28915652 0.27847876 0.27282583 0.28066699 
##        386        387        388        389        390        391        392 
## 0.25384089 0.25384089 0.28759639 0.33577802 0.30246826 0.31606367 0.32732901 
##        393        394        395        396 
## 0.29292514 0.29292514 0.33150286 0.37615900
```

```r
mse.test.covid <- mean((covid.test.tbl$deaths.100k-test.pred.covid)^2)
mse.test.covid
```

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

[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.

```r
calc_MSE_lag <- function(time.lag, train.tbl, test.tbl) {

  experimentTrain.tbl <- train.tbl %>%
    arrange(desc(date)) %>%
    mutate(deaths.100k = lag(deaths.100k, n = time.lag)) %>%
    filter(is.na(deaths.100k) == FALSE)


  experimentTest.tbl <- test.tbl %>%
    arrange(desc(date)) %>%
    mutate(deaths.100k = lag(deaths.100k, n = time.lag)) %>%
    filter(is.na(deaths.100k) == FALSE)

  linear.model <- lm(deaths.100k~cases.100k, data=experimentTrain.tbl)

  linear.model

  test.pred.covid <- predict(linear.model, experimentTest.tbl)

  test.pred.covid

  mse.test.covid <- mean((experimentTest.tbl$deaths.100k-test.pred.covid)^2)

  mse.test.covid
}

calc_MSE_lag(7, covid.train.tbl, covid.test.tbl)
```

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

```r
calc_MSE_lag(14, covid.train.tbl, covid.test.tbl)
```

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

```r
calc_MSE_lag(21, covid.train.tbl, covid.test.tbl)
```

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

calc_MSE_lag(7, covid.train.tbl, covid.test.tbl) [1] 0.02200948 calc_MSE_lag(14, covid.train.tbl, covid.test.tbl) [1] 0.01464344 calc_MSE_lag(21, covid.train.tbl, covid.test.tbl) [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.

```r
mseVectorCovid <- function(startVal, endVal) {
  vector = c()
  for(i in seq(from=startVal, to=endVal, by=1)){
  vector[i] = calc_MSE_lag(i, covid.train.tbl, covid.test.tbl)#/7
  }
  vector
}
testMseCovidVector <- mseVectorCovid(0,31)
testMseCovidVector
```

```
##  [1] 0.03502926 0.03262704 0.03015861 0.02775343 0.02561015 0.02363730
##  [7] 0.02200948 0.02061036 0.01930345 0.01793391 0.01669563 0.01577624
## [13] 0.01504106 0.01464344 0.01449611 0.01446863 0.01425157 0.01430079
## [19] 0.01470437 0.01535475 0.01617032 0.01719733 0.01833959 0.01933646
## [25] 0.02060774 0.02227927 0.02408032 0.02606224 0.02822873 0.03058582
## [31] 0.03288742
```

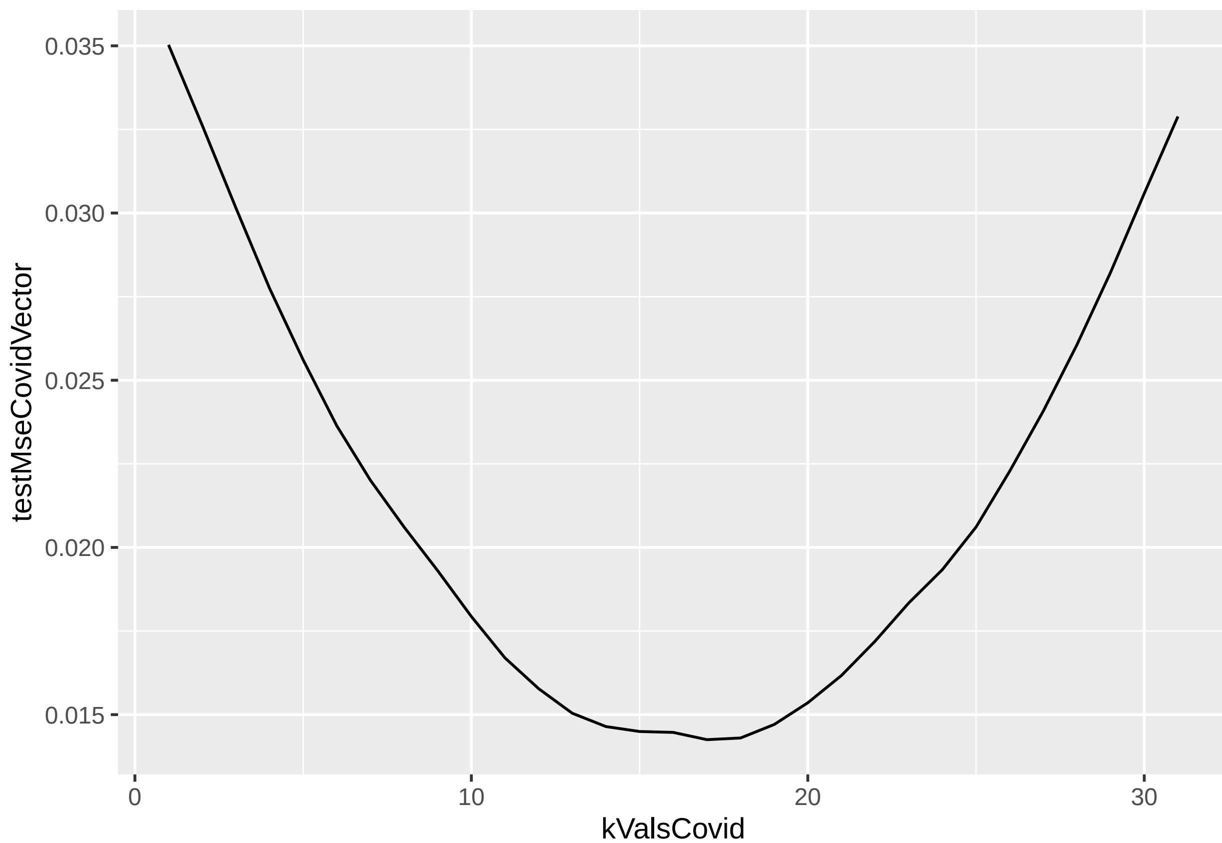
generate graph:

```r
kValsCovid = c()
for(i in seq(from=0, to=31, by = 1)){#, by=7
  kValsCovid[i] = i#[i/7]
}
kValsCovid
```

```
##  [1]  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] 26 27 28 29 30 31
```

get into tibble for plotting:

```r
covidTibble <- as_tibble(data.frame(kValsCovid, testMseCovidVector))
covidTibble %>%
  ggplot(mapping = aes(kValsCovid, testMseCovidVector)) +
  geom_line(mapping = aes(kValsCovid, testMseCovidVector))
```




```r
covidTibble
```

```
## # A tibble: 31 x 2
##    kValsCovid testMseCovidVector
##         <dbl>              <dbl>
##  1          1             0.0350
##  2          2             0.0326
##  3          3             0.0302
##  4          4             0.0278
##  5          5             0.0256
##  6          6             0.0236
##  7          7             0.0220
##  8          8             0.0206
##  9          9             0.0193
## 10         10             0.0179
## # ... with 21 more rows
```

```r
covidTibble %>%
  slice_min(testMseCovidVector, with_ties = FALSE)
```

```
## # A tibble: 1 x 2
##   kValsCovid testMseCovidVector
##        <dbl>              <dbl>
## 1         17             0.0143
```

kValsCovid testMseCovidVector 1 17 0.0143

The optimal lag time predicted by the KNN model is 17 days (between the number of cases to the number of deaths that follow).