MA615 Final Project

Kaiwei Xiao

2022-12-17

```r
library(ggplot2)
library(dplyr)
library(svglite)
library(devtools)
library(forecast)
library(tseries)
library(ggplot2)
library(ggfortify)
library(fpp)
library(vars)
```

```r
#2022 Quarter 1 Green lines travel data
Q1L = read.csv("2022-Q1_LRTravelTimes.csv")
```

```r
##2022 Quarter 1 other lines travel data
Q1H = read.csv("/Users/j/MA615/final project/TravelTimes_2022/2022-Q1_HRTravelTimes.csv")

Q2L = read.csv("/Users/j/MA615/final project/TravelTimes_2022/2022-Q2_LRTravelTimes.csv")

Q2H = read.csv("/Users/j/MA615/final project/TravelTimes_2022/2022-Q2_HRTravelTimes.csv")

Q3L = read.csv("/Users/j/MA615/final project/TravelTimes_2022/2022-Q3_LRTravelTimes.csv")

Q3H = read.csv("/Users/j/MA615/final project/TravelTimes_2022/2022-Q3_HRTravelTimes.csv")
```

```r
# Append the quarter 1 dataset
Q1 = rbind(Q1H, Q1L)
head(Q1)
```

```
##   service_date from_stop_id to_stop_id route_id direction_id start_time_sec
## 1   2022-01-01        70004      70001   Orange            0          45822
## 2   2022-01-01        70004      70001   Orange            0          45355
## 3   2022-01-01        70004      70001   Orange            0          58824
## 4   2022-01-01        70004      70001   Orange            0          57792
## 5   2022-01-01        70004      70001   Orange            0          44557
## 6   2022-01-01        70004      70001   Orange            0          44187
##   end_time_sec travel_time_sec
## 1        46027             205
## 2        45556             201
## 3        59108             284
## 4        58021             229
## 5        44776              219
## 6        44396              209
```

```
#check how many lines in this dataset
levels(factor(Q1$route_id))
```

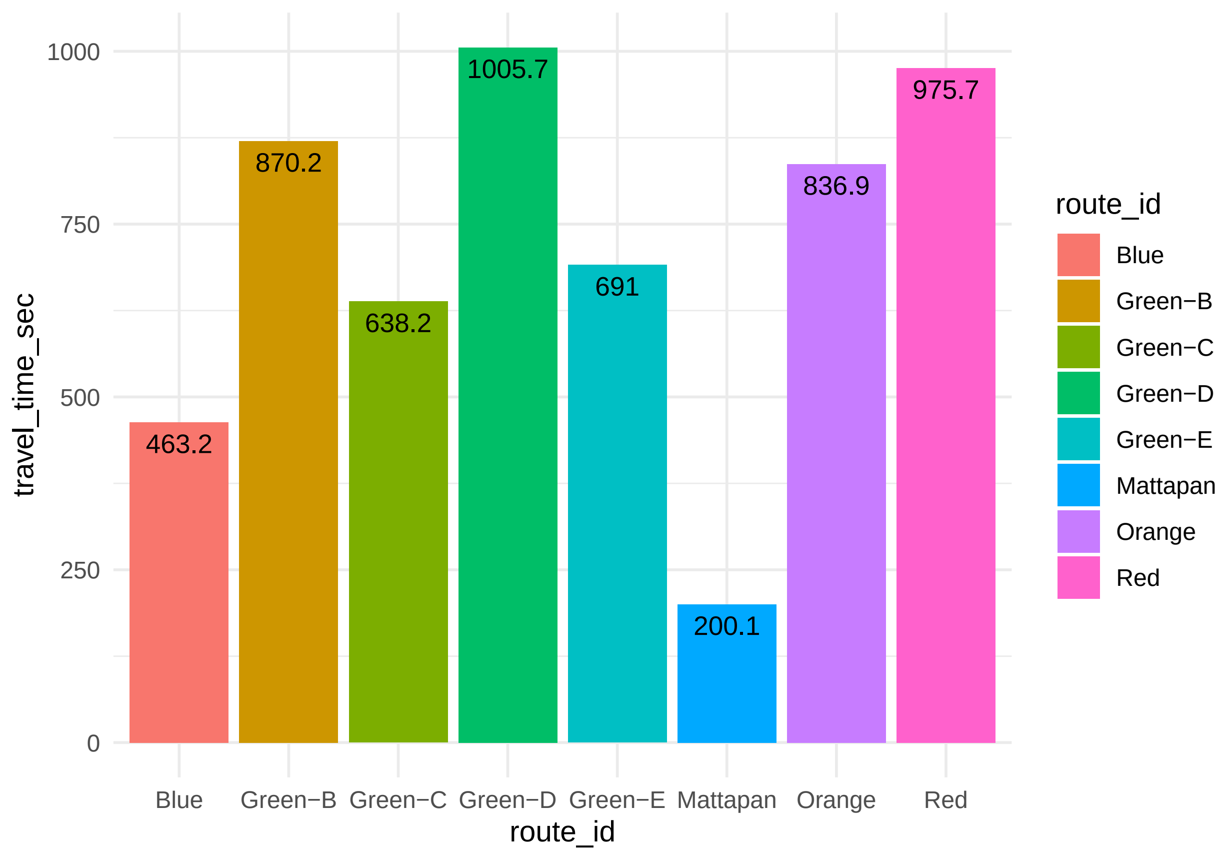
```
## [1] "Blue"     "Green-B"  "Green-C"  "Green-D"  "Green-E"  "Mattapan" "Orange"
## [8] "Red"
```

There are couple lines here.Green(including subset), Blue, Orange and Red lines. There is also a line called "Mattapan".

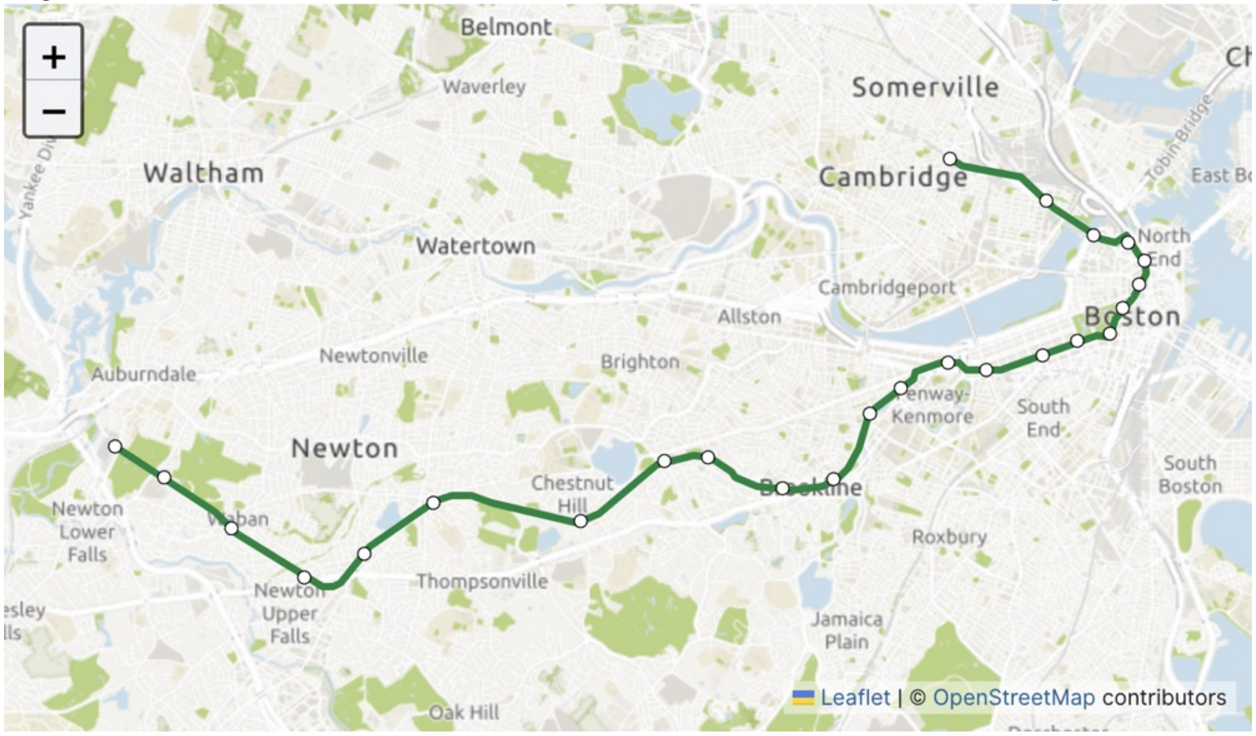
```
#compare average travel time of different lines
travelt = aggregate(travel_time_sec ~ route_id, data = Q1, mean)
travelt$travel_time_sec = round(travelt$travel_time_sec, digits = 1)
travelt
```

```
##   route_id travel_time_sec
## 1     Blue           463.2
## 2  Green-B           870.2
## 3  Green-C           638.2
## 4  Green-D          1005.7
## 5  Green-E           691.0
## 6 Mattapan           200.1
## 7   Orange           836.9
## 8      Red           975.7
```

```
ggplot(data = travelt, aes(x = route_id, y = travel_time_sec, fill = route_id)) +
  geom_bar(stat = "identity")+
  geom_text(aes(label= travel_time_sec), vjust=1.6, color="black", size=3.5)+
  theme_minimal()
```



As we can see, the Mattapan has the shortest average travel time. The Green-D line has the longest average travel time which is 1005.7 seconds. It is almost 16.7 minutes for it to travel from a stop to the next one!

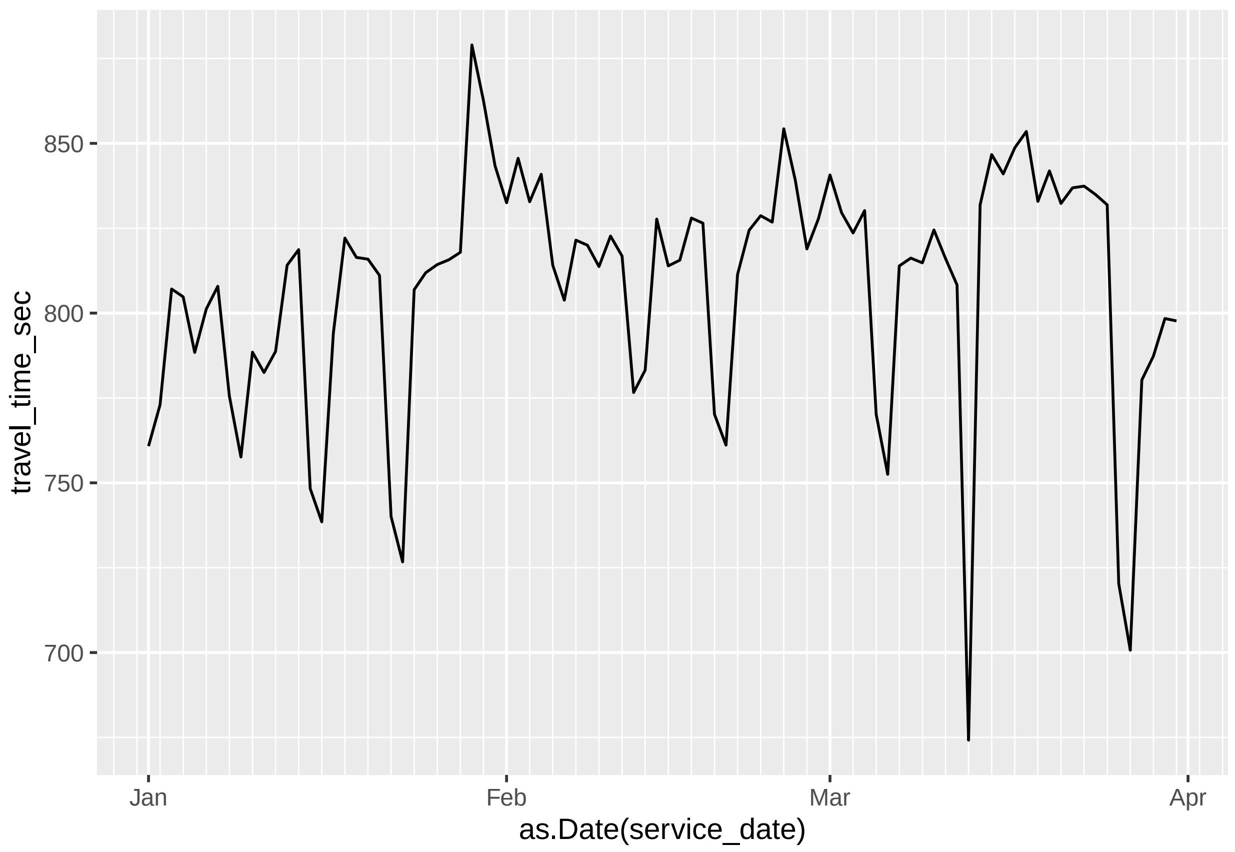


It is very reasonable because as you can see, the Green line D is a long line, and the distance from stop to stop is quite long.

```
#Let's see how travel time impacted by the date.
traveld = aggregate(travel_time_sec ~ service_date, data = Q1, mean)
traveld$travel_time_sec = round(traveld$travel_time_sec, digits = 1)
head(traveld)
```

```
##   service_date travel_time_sec
## 1   2022-01-01           760.8
## 2   2022-01-02           773.0
## 3   2022-01-03           807.1
## 4   2022-01-04           804.8
## 5   2022-01-05           788.4
## 6   2022-01-06           801.2
```

```
#Plot average travel time vs date
p = ggplot(traveld, aes(x=as.Date(service_date), y=travel_time_sec)) +
  geom_line()
p + scale_x_date(date_minor_breaks = "2 day")
```



```
#see which date got the longest mean travel time.
traveld[which.max(traveld$travel_time_sec),]
```

```
##    service_date travel_time_sec
## 29   2022-01-29             879
```

```r
#see which date got the shortest mean travel time.
traveld[which.min(traveld$travel_time_sec),]
```

```
##    service_date travel_time_sec
## 72   2022-03-13           674.2
```

It seems that in the quarter 1 time range, the travel time trend is not that obvious. How about looking through the 2022 year?

```r
#combine the 2022
QA = rbind(Q1H, Q1L, Q2H, Q2L, Q3H, Q3L)
head(QA)
```

```
##   service_date from_stop_id to_stop_id route_id direction_id start_time_sec
## 1   2022-01-01        70004      70001   Orange            0          45822
## 2   2022-01-01        70004      70001   Orange            0          45355
## 3   2022-01-01        70004      70001   Orange            0          58824
## 4   2022-01-01        70004      70001   Orange            0          57792
## 5   2022-01-01        70004      70001   Orange            0          44557
## 6   2022-01-01        70004      70001   Orange            0          44187
##   end_time_sec travel_time_sec
## 1        46027             205
## 2        45556             201
## 3        59108             284
## 4        58021             229
## 5        44776             219
## 6        44396             209
```

```r
#let's see if there is any new lines appear.
levels(factor(QA$route_id))
```

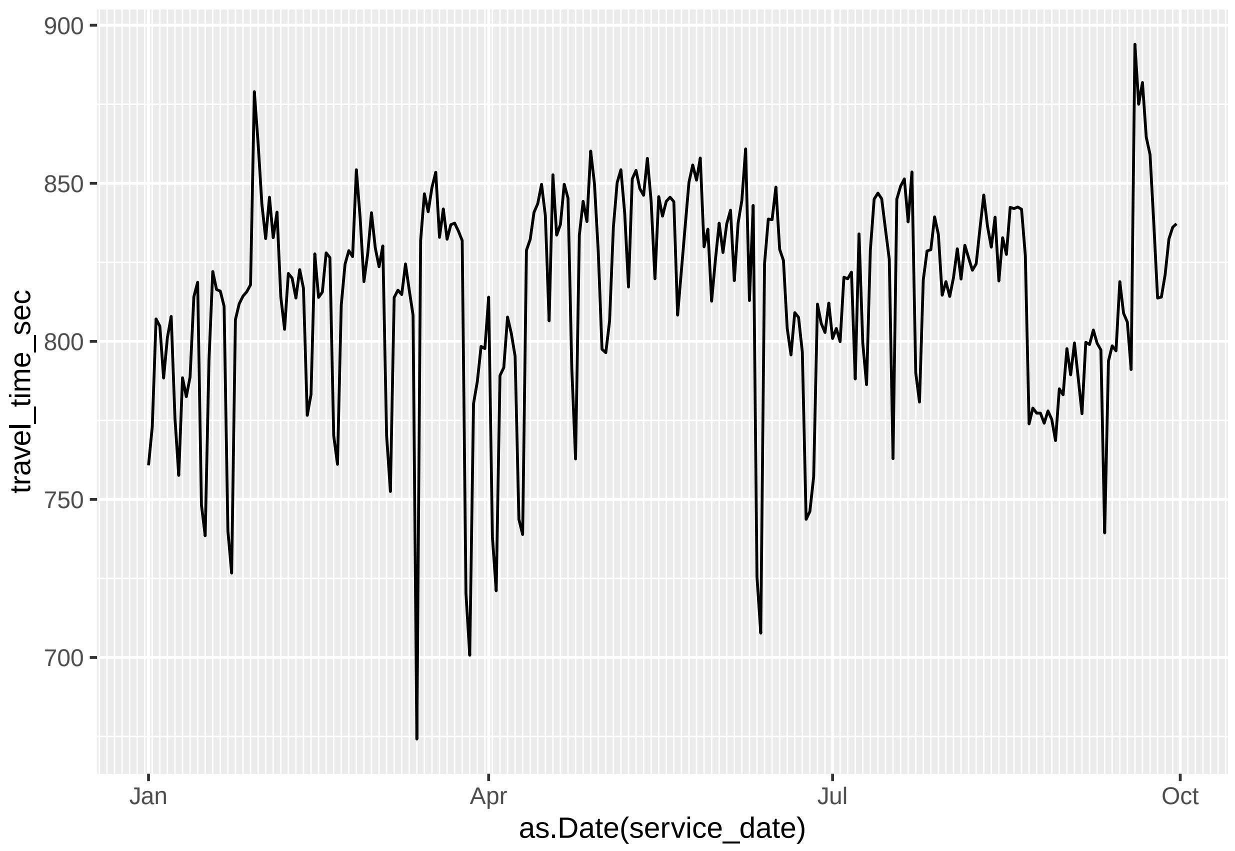
```
## [1] "Blue"     "Green-B"  "Green-C"  "Green-D"  "Green-E"  "Mattapan" "Orange"
## [8] "Red"
```

Still, we got the same line in the rest of the year.

```r
#let's check the travel time by date through 2022
travelda = aggregate(travel_time_sec ~ service_date, data = QA, mean)
travelda$travel_time_sec = round(travelda$travel_time_sec, digits = 1)
head(travelda)
```

```
##   service_date travel_time_sec
## 1   2022-01-01           760.8
## 2   2022-01-02           773.0
## 3   2022-01-03           807.1
## 4   2022-01-04           804.8
## 5   2022-01-05           788.4
## 6   2022-01-06           801.2
```

```r
#Plot average travel time vs date
pa = ggplot(travelda, aes(x=as.Date(service_date), y=travel_time_sec)) +
  geom_line()
pa + scale_x_date(date_minor_breaks = "2 day")
```



```r
#see which date got the shortest mean travel time.
travelda[which.min(travelda$travel_time_sec),]
```

```
##    service_date travel_time_sec
## 72   2022-03-13           674.2
```

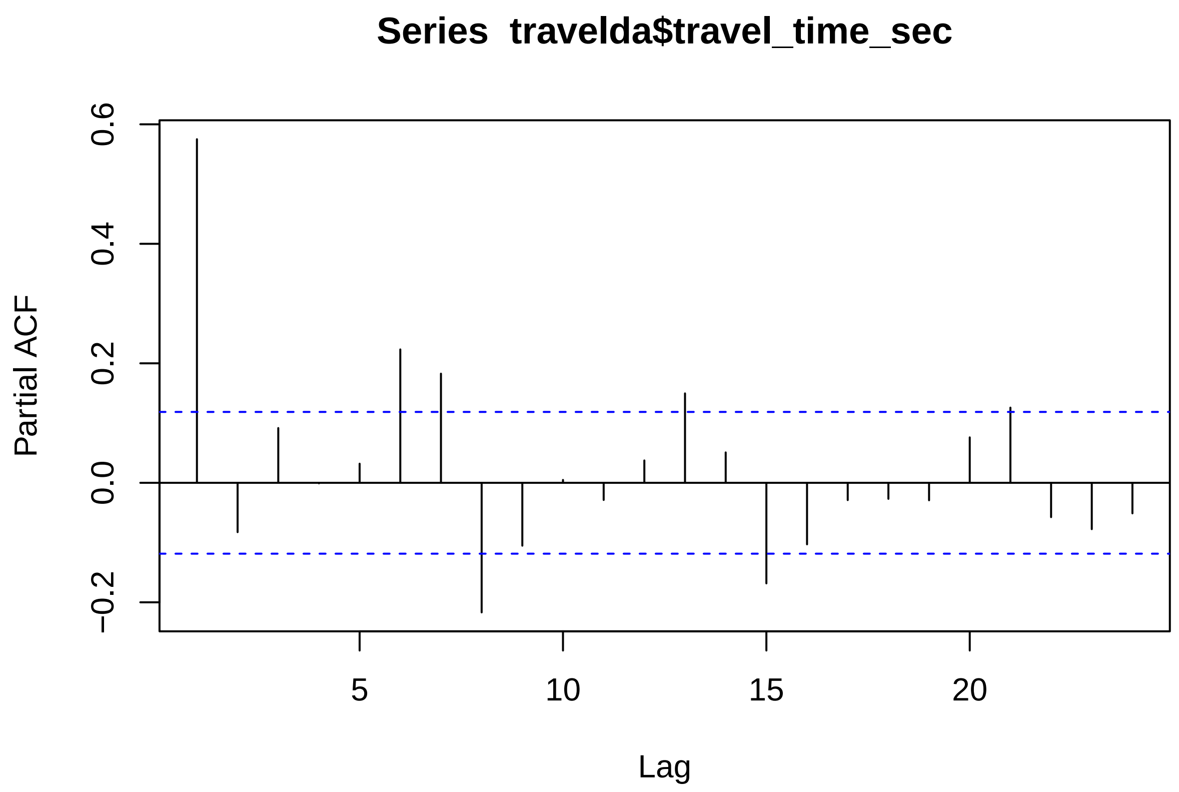
```r
#see which date got the longest mean travel time.
travelda[which.max(travelda$travel_time_sec),]
```

```
##     service_date travel_time_sec
## 262   2022-09-19             894
```

The shortest mean travel time happened on March 13,2022 and it was 674.2 sec. The longest mean travel time happened on September 19,2022 and it was 894 sec.

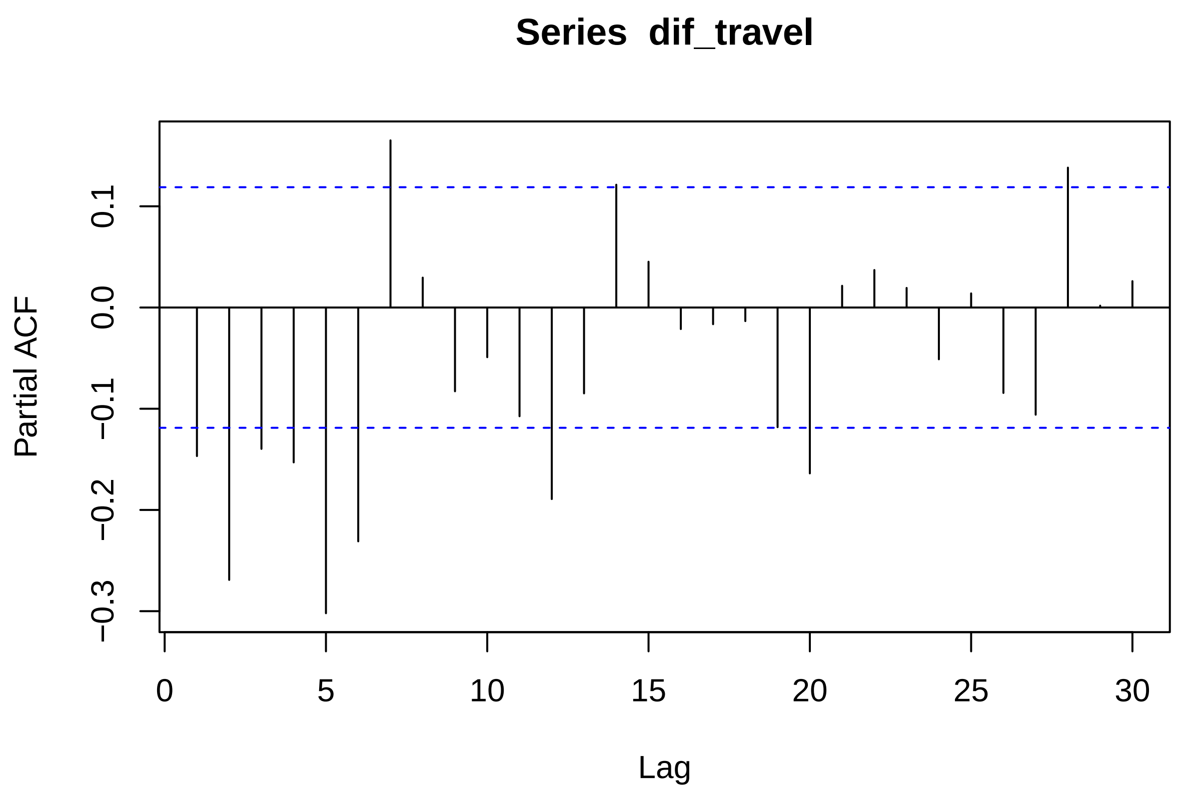
Can we build a time series model for the travel time?

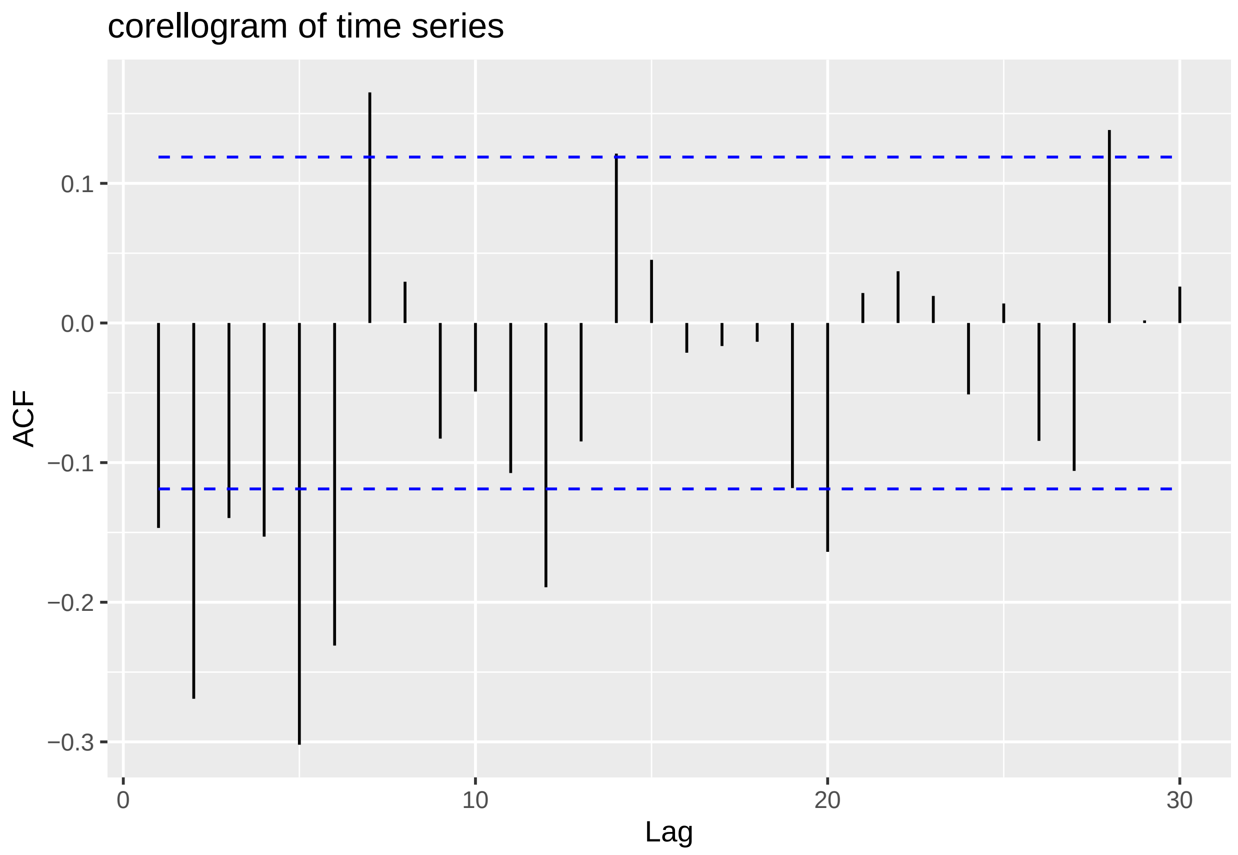
```
pacf(travelda$travel_time_sec)
```



```
#take difference for time seiries model building
dif_travel = diff(travelda$travel_time_sec)
```

```
#plot the time series data
autoplot(
  pacf(dif_travel, lag.max = 30),
  main="corellogram of time series"
)
```


Series dif_travel
Partial ACF
Lag
0.1
0.0
−0.1
−0.2
−0.3
0
5
10
15
20
25
30



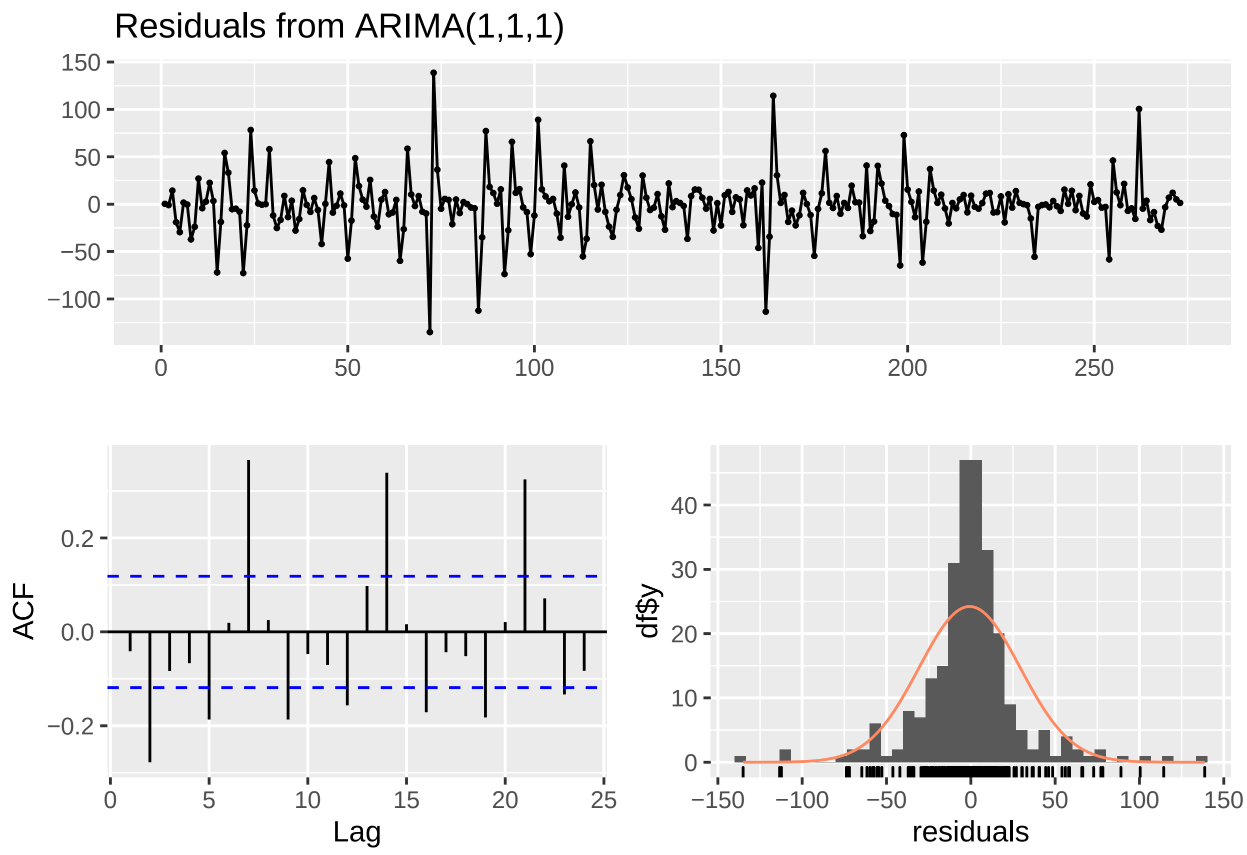
As an inital pass, we will fit ARIMA(1,1,1) with seasonal difference.

```
model_arima <- Arima(
  y = travelda$travel_time_sec,
  order = c(1, 1, 1),
  seasonal = list(order = c(0, 1, 0))
)

model_arima
```

```
## Series: travelda$travel_time_sec
## ARIMA(1,1,1)
##
## Coefficients:
##           ar1      ma1
##       -0.1432  -1.0000
## s.e.   0.0601   0.0093
##
## sigma^2 = 912.3:  log likelihood = -1310.03
## AIC=2626.06   AICc=2626.15   BIC=2636.87
```

```
checkresiduals(model_arima)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1)
## Q* = 83.383, df = 8, p-value = 1.021e-14
##
## Model df: 2.   Total lags used: 10
```

The Ljung-Box test test the overall randomness of the time seiries. H0: The data are independently distributed (i.e. the correlations in the population from which the sample is taken are 0, so that any observed correlations in the data result from randomness of the sampling process). Ha: The data are not independently distributed; they exhibit serial correlation.

Here the test p-value is significant, we can reject the null and conclude that the data are not independently distributed, there is serial correlation here.

Now we use auto.arima to see what we get using Hyndman & Khandakar (2008) algorithm. This algorithm can help us choose the p and q for the ARIMA model based on AIC.

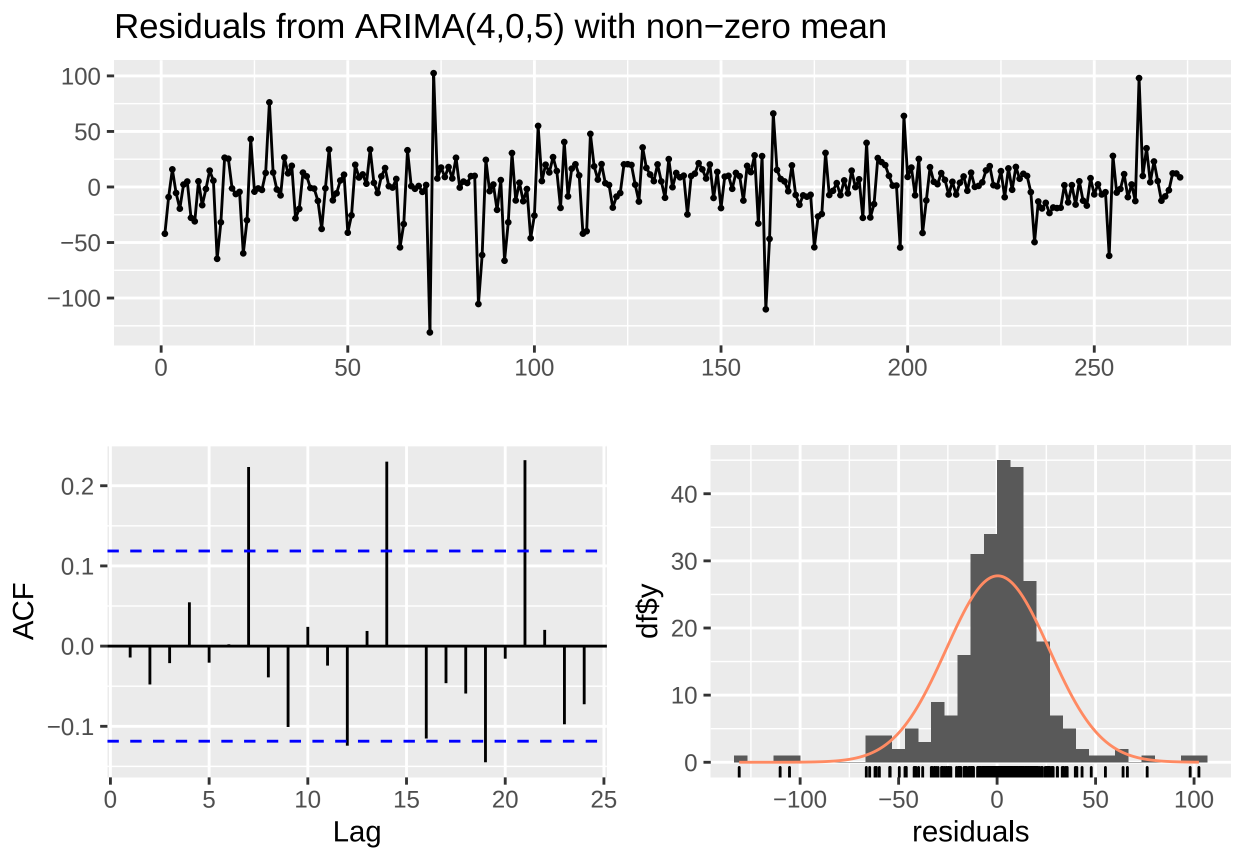
```
sarimax_petro_law = auto.arima(
  y = travelda$travel_time_sec,
  ic = "aic",
  max.order = 10,
  stepwise = F,
  approximation = F,
  parallel = T,
  num.cores = 4
)

# best model
sarimax_petro_law
```

```
## Series: travelda$travel_time_sec
## ARIMA(4,0,5) with non-zero mean
##
## Coefficients:
##          ar1      ar2     ar3     ar4      ma1     ma2     ma3      ma4
##       1.1742  -0.6868  0.0564  0.3056  -0.5563  0.2828  0.2439  -0.3130
## s.e.  0.4956   0.8361  0.7584  0.3525   0.4910  0.5456  0.4087   0.1043
##          ma5      mean
##       -0.1696  814.5509
## s.e.   0.1610    5.0334
##
## sigma^2 = 707.5:  log likelihood = -1278.36
## AIC=2578.72   AICc=2579.73   BIC=2618.42
```

It choose p = 4 and q = 5 with no difference(I=0) for the model automatically. Also we can see the all year mean travel time for all lines is 814.55 sec.

```
checkresiduals(sarimax_petro_law)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(4,0,5) with non-zero mean
## Q* = 23.948, df = 3, p-value = 2.561e-05
##
## Model df: 9.   Total lags used: 12
```

The Ljung-Box test tell us there is serial correlation here and the model is meaningful.

We can write the model in equation here: Average_travel_time = 814.6+1.17Y(t-1)-0.69Y(t-2)+0.06Y(t-3)+0.31Y(t-4) -0.56e(t-1)+0.28e(t-2)+0.24e(t-3)-0.31e(t-4)-0.17(t-5)+error

The Y(t) means the p days average travel time in front of the date you want to predict, and the e(t) means the white noise of series in front of the date you want to predict and the error is the error!

So if you want to predict the average travel time by date, use this model.

##Shiny app

for the shiny app there are two version

The first version is driving map, which allow you to add waypoint on it. Simply type in the departure address and final destination (waypoints if needed), you can get the driving routine. Also use your map key and API key if possible.

#published link: https://jpz61m-jeremy-x0.shinyapps.io/Driving_Map/

The second version is transit map. It allows you travel through public transportation but no waypoints due to the API limitation.

#published link: https://jpz61m-jeremy-x0.shinyapps.io/Transit_Map/